Exploration by Unsupervised Imitation Learning

Denis Pogosov
Radboud University Nijmegen
denis.b.pogosov@gmail.com

Abstract

Deriving of a reward is a major challenge in reinforcement learning frameworks under the natural environment settings. The natural environment not only tends to crucially delay reinforce but also has sparse continuous search space with limited search time. To overcome these problems, we investigated unsupervised imitation learning. We have shown experimentally that our approach improves performance of the models based on (un)directed exploration and the performance was significant. Moreover, unsupervised imitation learning does not require to query and execute expert, and even does not require to designate an expert explicitly.

1 Introduction

Reinforcement learning (RL) unites biological and technical concepts for solving a broad class of control problems. Exploration plays a fundamental role in RL systems. RL agents probe an area of the search space to incrementally update their learnable parameters. There exists a variety of research regarding exploration in RL. Exploration techniques can be divided into two broad categories [1]: (un)directed exploration and imitation learning.

Undirected exploration [1, 2] is the simplest techniques and it is based on random search, which do not take into account the history of learning process. It is pure random (uniformly distributed), semi-uniform distributed and Boltzmann distributed explorations. For example, a popular RL approach [3] incorporates Ornstein-Uhlenbeck process, that is actually decayed random search. A massive variant of random search is realized in asynchronous actor-critic algorithms [4] and bootstrapping [5] algorithms, where better samples were taken from parallel (still random) interactions with an environment. Experience replay [6] is an inherent component of the exploration in recent RL methods. Prioritized experience replay [7] optimizes taking traces from replay memory to speed up learning from the newly explored traces with a higher temporal-difference error.

Directed exploration techniques consider some heuristics in the learning history in order to affect the locations of the environment that should be explored. For instance, Counter-based exploration considers the states on the basis of how often they visited. Counter/Error-based exploration [8], [9] incorporates the changes in the state utility in order to make more frequent occurring the states which greater utility. Recency-based exploration [10] makes the states that were visited recently less preferable. We accentuate that Directed exploration benefits from the states that were previously explored; they do not optimize searching of new states with a higher reward.

Imitation learning (IL) is a different approach that hinges on the behavior of an expert. We emphasize two categories: supervised and unsupervised IL. Supervised imitation learning exploits an expert policy, therefore an expert has to be designated explicitly. For instance, Forward training algorithm [11] trains a non-stationary policy iteratively in order to eliminate the regret of naive supervised learning. Stochastic Mixing Iterative Learning (SMILe) [11]) can be applied for a large number of iterations and trains a stationary policy. Both these methods require the expert policy. DAGGER [12] is an iterative algorithm that trains a single deterministic policy and remembers the experience of the expert and previous policies. This approach queries an expert to interact with trajectories directly.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

One shot imitation learning [13] offers a framework for selected problems, which do not require task-specific engineering and multiple sampling. This approach exploits DAGGER and still requires access to the expert policy.

An ideal environment provides immediate rewards, therethrough allows agents to learn continuously. However, the natural environment not only tends to crucially delay reinforce but also has sparse continuous search space with limited search time. Both these issues make reinforce almost unachievable by (un)directed exploration techniques. Furthermore, any access to the expert policy is unnatural and biologically implausible. That is why the supervised IL exploration approaches could be inefficient for the real-world-affine problems and reproduce neither animal nor human abilities to explore.

In this paper, we investigate an approach to unsupervised imitation learning, where an efficient learner is able to explore on the observed trajectories of prospective experts and does not require their policies.

2 Materials and Methods

Without loss of generality we consider exploration by unsupervised imitation learning on the example of locomotion control framework.

2.1 Concept of unsupervised imitating learning

Continuously interacting moving objects (e.g. road users) make the natural environment essentially dynamic. An RL agent should predict a future state of the environment to act efficiently. Furthermore, interactions of other moving objects with the environment is a vast source of exploration experience. Our approach incorporates both these abilities. The intuition here is that under uncertainty conditions an RL agent samples a possible action from the behavioral statistics of other moving objects with lower uncertainty, pro rata to behavioral resemblance measure. We emphasize that all the objects may act as experts. An object with stronger statistical correlation between the given input and the coupled behaviour as well as with higher resemblance is more likely to "act" as an expert.

We model the prediction of other objects behavior as well as the RL agent actions by Gaussian functions, which represent position and orientation. This leads to a short-term planning framework. Hence, our approach is beyond an end-to-end neural network. However, the external motion controller, which is required in planning settings, is devised with the nonlinear control theory, which benefits us to a greater degree. We can guarantee the fundamental properties of the control system such as stability and controllability [14] that are difficult [15] for neural networks. Furthermore, our approach represents the perception-action-control cognitive cycle. Outline of unsupervised imitation learning is shown on the Figure 1, which we are about to explain.

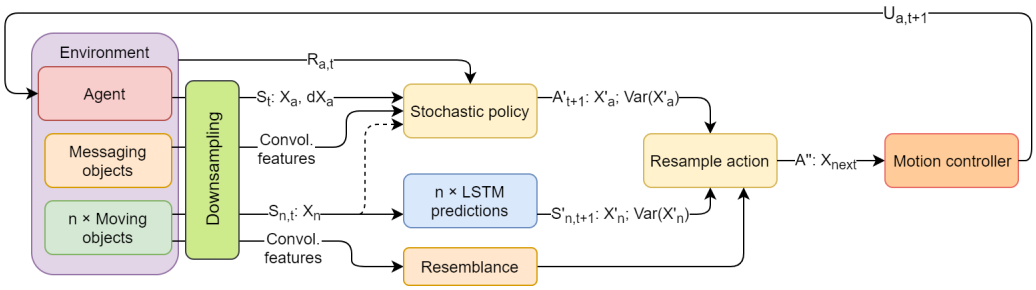

Figure 1: Outline of unsupervised imitation learning.

Solving complex problems from high-dimensional, e.g. raw pixel, sensory input is a primary task of an efficient RL agent under unsupervised conditions. Locomotion control handles with the position, orientation($S_t : X_a$) and their derivatives ($S_t : dX_a$) of the (controlled) objects. To get them, the high-dimensional sensory input should be reduced. For the controlled object it is convenient to use [16] localization by inertial navigation, visual odometry and so on. For the other moving objects, raw pixel input is segmented by a convolutional network into a number n of objects, each belongs to a certain class and has position coordinates ($S_{n,t}$). Both these two operations are united under

Downsampling step on the Figure 1. On the one hand, *LSTM predictor* traces each of the detected moving object. Afterwards, the predictor estimates the future state of the environment on base of locomotion statistics with the shared class properties. Further, the predictor estimates the future state of the moving objects $S'_{n,t+1}$ (how do they behave) if they take the same input as the agent. On the other hand, *Stochastic policy* under the reinforcement learning framework outputs an action A'_{t+1} with its variance given the predicted state of the environment. We consider the action variance and the predictions variance as a measure of uncertainty. At *Resample action* step, under the action uncertainty conditions, a new action *A"* is sampled from the predictions of other objects taking into account their behavioral resemblance to the agent. Finally, a motion controller performs the new sampled action ($U_{a,t+1}$).

We consider these components of unsupervised imitation learning framework in further detail.

2.2 Classification of moving objects from raw pixel input

We propose segmentation of raw pixel input by Mask R-CNN convolutional neural network [17]. This approach is substantively based on Faster R-CNN convolutional network [18]. Both they have a set of convolution layers that outputs Regions of Interests (RoI) to further classification and regression. Alongside the regression branch (predicts an object bounding box) and classification branch (predicts an object class) Mask R-CNN has a masking branch for pixel-wise classification, which uses the same RoI among all the branches. In the man-made environment a RL agent should recognize the meaning of messaging objects, such as road signs, traffic lights, information boards, etc. For that purpose, Mask R-CNN first detects messaging objects. Second, high-order convolutional features, which are belong to the messaging objects, are reshaped and passed to *Stochastic policy*.

2.3 Prediction in dynamic environment

Formalization or approximation of motion reasoning is very difficult. However, shared reasoning occurs frequently in driving settings, where a local group follow similar rules or aims. There are strong statistical dependencies in mobile objects behavior under certain scenarios such as reaction to an unidentified situation (e.g. unknown road sign), obstacle avoidance, etc. The samples from such behavior are assumed independent and identically-distributed. According to the central limit theorem we can model these behavioral dependencies by Gaussian functions having sufficient statistical data. The quality of predictions depends on sufficient amount of the statistical data because the prediction is purely statistically driven. The sufficiency can be represented by a number of trajectory samples that predictor should take for learning itself. Furthermore, it is not necessary to specify the sufficiency by a quanta. If the data is not sufficient, the unsupervised imitation learning consistently explores undirectly (a random search). Further, at the prediction stage, the predictor takes observed motion statistics of the detected mobile objects to estimate their future state.

We approximate long-time statistical dependencies by LSTM modelling. LSTM is proven to be efficient in human motion modelling [19] and target tracking [20]. Shared reasoning can be modelled by social pooling [19] that is the pooling of hidden LSTM states among neighboring objects (e.g. people), which share similar reasoning (due to possible interactions) and surroundings:

$$H_t^i(m,n,:) = \sum_{j \in N_i} 1_{mn}[x_t^j - x_t^i, y_t^j - y_t^i] h_{t-1}^j$$

where H_t^i is the hidden-state tensor, h_{t-1}^j is the hidden state of the LSTM corresponding to the *j* object at *t-1*, $1_{mn}[x, y]$ is an indicator function to check if *(x, y)* is in the *(m,n)* cell of the grid, and N_i is the set of neighbors corresponding to object *i*.

We propose to use one LSTM per class of the detected object. In the case of detection of several objects of the same class are detected, we use LSTM with shared weights and pooled hidden states. The agent and the detected mobile objects have different input space. Hence, we should use two different inputs: one for training and prediction of moving objects behavior; and the other for the agent exploration. In the prediction case, the input space is sliced for every detected object taking into account their visual field. In the exploration case, LSTM gets the whole visual scene. Consequently, the predictor estimates how the detected mobile objects would behave in the similar to the RL agent scenery.

We model the agent and other moving objects (enumerated with *i*) state by multivariate Gaussian with the mean $s_i^t = [x\ y\ dx\ dy\ v]^T$ and variance V_i, where x, y are position coordinates of the agent (or an moving object) center of mass in the fixed reference frame, dx, dy are their derivatives, and v is pseudovelocity. Similar state modeling for the agent and other objects allows to use predictions as exploration source.

2.4 Stochastic policy

We propose to model the stochastic policy by Stochastic Value Gradients (SVG) (0) method [21] that outperforms [22] deterministic policy gradients. In addition, SVG(0) is model-free and similar to deep deterministic policy gradient (DDPG), which can be used for benchmarking. Though SVG does not perform well in long-term planning [21], we can still apply it for short-term planning in our locomotion framework.

SVG exploits advantages of the re-parameterization of distributions. This allows to obtain a simple Monte-Carlo estimator of the derivative of an expectation. The core of SVG is backpropagation through the stochastic Bellman equation (with its derivatives)[21]:

$$V(s) = \mathbb{E}_{\rho(\eta)}\Big[r(s, \pi(s, \eta; \theta)) + \gamma \mathbb{E}_{\rho(\xi)} V'_{s'}(f_s + f_a \pi_s)\Big],$$

$$\frac{\partial V}{\partial s} = \mathbb{E}_{\rho(\eta)}\Big[r_s + r_a \pi_s + \gamma \mathbb{E}_{\rho(\xi)} V'_{s'}(f_s + f_a \pi_s)\Big],$$

$$\frac{\partial V}{\partial \theta} = \mathbb{E}_{\rho(\eta)}\Big[r_a \pi_\theta + \gamma \mathbb{E}_{\rho(\xi)}[V'_{s'} f_a \pi\theta + V'_\theta]\Big],$$

where $a = \pi(s, \eta; \theta)$ is the stochastic policy, $s' = f(s, a; \xi)$ is the stochastic environment. Lower index means partial derivative $X_y = \partial X / \partial y$. The tick notation marks a variable at the next time step. Other variables are common under a discrete-time Markov Decision Processes (MDPs) with continuous states and actions [21].

Then the gradient is evaluated on real trajectories. SVG(0) is a stochastic analogue of the recognized deterministic policy algorithm of DDPG [3] . SVG(0) assumes stochastic policy and estimates the derivative around the policy noise $\mathbb{E}_{p(\eta)}[Q_a \pi | \eta]$. This allows to learn policy noise variance, which we interpret as an action uncertainty.

2.5 Action resampling

At this moment the RL agent has the distribution of action (desired future state) and variance by the stochastic policy as well as the distributed predicted future state and variance of other moving objects. From these two distributions the agent will sample the new action *A"* (Figure 1) to perform.

Resemblance measure The action from the predictions of other objects cannot be taken directly, because their behavior may differ from the agent behavior. To deal with this issue, we introduce the measure of behavioral resemblance. Further the resemblance will be taken as probability to form the joint distribution with the predictions. One of the simplest and efficient [23] assumption is that resemblance in object structure determines resemblance in behavior. We also assume that object structure is encoded as a combination of high-order convolutional features, which represent different (functional) parts of an object. Therefore, the resemblance measure is calculated as variance with mean that corresponds the *class 0*, which describes the agent:

$$S_j = \mathbb{E}\Big[\sum_{i=1} (W_{ij} - W_{i0})^2\Big], j = 1...m,$$

where *W* is the vector of weights at fully-connected layer, *j* is an output unit (class) index, *i* is an input unit index, *n* is the number of input units, *m* is the size of resemblance vector *S*.

Sampling behaviour from all the kind of moving objects is not reasonable. Some of them have very different behaviour and cannot be useful in any case. Hence, we limit the size of vector *S* only by the classes that have some threshold resemblance value S_{th} (e.g. larger than 0.7).

Double sampling We propose two variants of *Action resampling*. Both consist of two steps. At the first (sampling) step *Stochastic policy* (SVG(0)) samples an action A' with probability $p_{svg} = Z \cdot V'$ (refer the Figure 1). The probability is normalized by Z to be valid. At the second (resampling) step the final action from the vector a is sampled with probability from the vector p:

$$p = \left[p_{svg} \quad \frac{(1 - p_{svg})}{\sum_{i=1}^{m} S_i V_i^{-1}} \vec{S} \circ \vec{V}^{-1} \right], \quad \sum_{i=1}^{m+1} p_i = 1$$

$$a = [A' \quad S_1 \quad ... \quad S_m],$$

where $\vec{S}$ is the resemblance vector and $\vec{V}$ is the vectorized variance of the predictions of other objects behaviour, $S_{1...m}$ is the predicted actions (Figure 1), $dim(p) = dim(a) = (m + 1) \times 1$.

The other variant is a "greedy" version of the second step action sampling:

$$p_{gr} = \left[1 - S_{max} V_{max}^{-1} (1 - p_{svg}) \quad S_{max} V_{max}^{-1} (1 - p_{svg}) \right]$$

$$a_{gr} = [A' \quad S_{p,max}],$$

where S_{max}, V_{max}, $S_{p,max}$ are resemblance, variance and the predicted state obtained by maximizing with respect to some criteria (e.g. S=1 or $min(V)$), $dim(p_{gr}) = dim(a_{gr}) = 2 \times 1$.

The agent's own experience is more significant because it is reinforced as distinct from other object behavioral statistics, which reinforce is not accessible for the agent. That is why the final action is resampled with the probability $1 - p_{svg}$ and does not resampled with e.g. softmax function. In view of this, the agent relays mainly on its own experience and resamples the action only under uncertainty conditions.

Mixing iterative learning In imitation learning settings the learned policy influences the future test states on which it is tested further. As a consequence, the training and testing data are not independent and identically-distributed. This leads to increase errors [11]. We introduce mixing iterative learning, which is similar to SMILe algorithm apart that our approach is not geometric mixing but more momentum:

$$A'' = A'(1 - \gamma^k) + S_j \gamma^k,$$

where γ is a decay rate (e.g. 0.99), k is the number of step which resets to zero if the action is newly resampled from another object behaviour prediction.

Training occurs over several episodes that allows the learner policy be slowly modified from executing another object behaviour to the learned policy. This significantly improves stability of learning.

We note that for safety reasons a new action should be limited (out of scope of this work) based on physical constraints, such as maximum speed (allowed by road conditions), being on the road, etc.

3 Results

We analyzed unsupervised imitation learning in a toy environment with the properties of delayed reinforce and sparse search space. For the toy problem we did not learn from pixels with downsampling. We assumed only one moving object with correct behavior and resemblance = 1 that we called the conductor object. As the Stochastic policy we selected Stochastic Value Gradients (0) algorithm. The outline of the unsupervised imitating learning under the toy environment settings is shown on the Figure 2.

3.1 Experiment setup

The aim of the RL agent in the toy environment was to reach the point (target). The RL agent learned by sampling from the predicted behavior of the conductor under uncertainty conditions. The conductor appeared one time per episode. The toy environment output is shown on the Figure 3.

The target is represented by the small circle. Green line is the trace of the conductor. The conductor starts from random point on the large circle. Red line is trace of the RL agent. Blue lines are predicted behavior of the conductor for every time step. Title shows: the number of episodes when the target

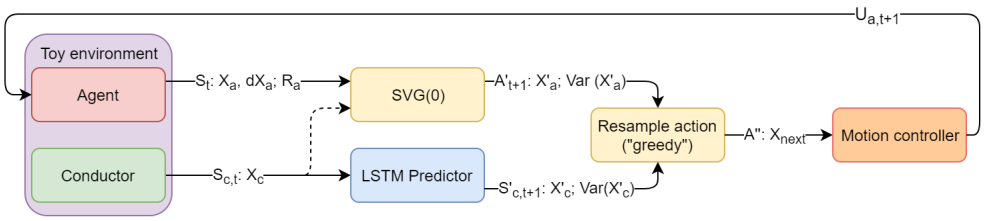

Figure 2: Outline of the unsupervised imitating learning under the toy environment.

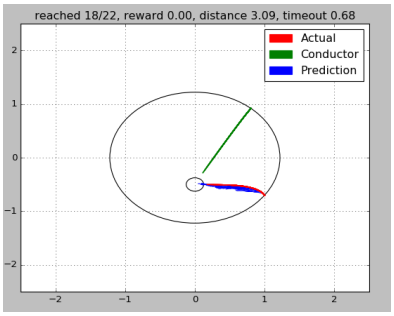

Figure 3: The toy environment.

was reached over the total number of episodes; immediate reward; distance to the target; and timeout (0∼1).

The toy environment outputted the position of the agent vector, its derivative, the position of the conductor ($S_{c,t}$) as well as a reward.

Reward was designed as delayed reinforce. Only when the target could be reached the RL agent received a reward: $r = 100 - \int dXdt + ||p_{target} - p_{start}||$, which was a positive number minus a penalty for a long path and the episode ended.

Search space was designed sparse. The search time was limited by 350 steps whereas the target cannot be reached faster than 200 steps from the starting point. Distance to the target was about 15% of one space dimension. These settings imitated extreme conditions where an RL agent had to react fast.

The motion controller was a simple proportional controller that performed locomotion from a current position to the desired point. The control action was limited to keep the steps small (the target could not be reached faster than 200 steps).

3.2 Analyzing of behavioral prediction

To generalize the behavior of the conductor object we simulated its behavior under the random settings: it started from a random position (belonged to a circle) and reached the same target as was designated for the RL agent. Therewith, we did not show to the RL agent what it had to do exactly by doping it with explicit expert trajectories. Resemblance to the conductor was 1.0 and class likelihood was 1.0. The predictor was LSTM one layer of 128 units. It converged (Figure 4) within 100 epochs (50 000 steps). Final error was at the level of 6.2. For this particular task we interpret the convergence numbers as the criteria of sufficiency of statistical data.

The samples of the predicted trajectories are shown on the Figure 5. Despite the fact that the trajectories were noisy, they simulated towards the target. Moreover, the agent used only the last predicted point (planning framework with locomotion control to a point). That is why we see its smooth motion. Hence, the generalized and therefore noisy and not error-free predictions are relevant for smooth and locomotion control towards the true target under a planning framework.

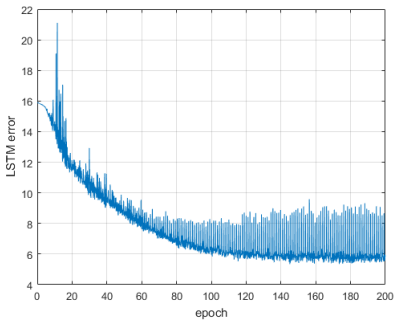

Figure 4: Convergence curve of LSTM predictor.

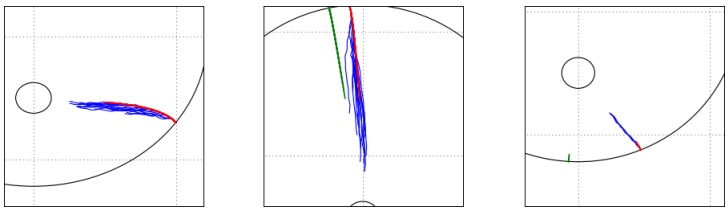

Figure 5: Samples of the predicted trajectories.

3.3 Analyzing of unsupervised imitation learning

We compared our approach (SVG(0)+LSTM perdictor) with SVG(0) and DDPG in the same environment. In the last case, the agent observation space was augmented with the conductor position coordinates (dashed line on the Figure 2) to simulate downsampled input of the all visual scene (in a similar way to [24, 3]).

Both actor and critic networks of SVG(0) and DDPG agents were two fully-connected layers of 100 units; replay buffer had size of 1000 (minimum 256); the batch had size of 64.

We made 20 *runs* of 350/500 episodes. An experimental run was considered solved if the agent reached the target within 10 episodes in succession without sampling from the LSTM predictor, i.e. based only on the learner policy only. Three examples of experimental run are shown on the Figure 6. The blue lines mean whether the run is solved or not; and the red ones indicate the proportion of steps were resampled from the predictions to the toal number of steps.

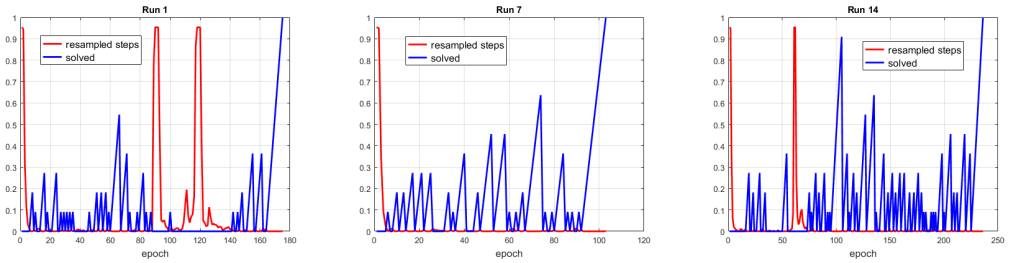

Figure 6: Experimentral runs.

We note unstable behaviour of SVG(0) from the Figure 6. Moreover, it could not converge with more than 15 episodes in succession within 500 epochs. Alongside this, it was prior tested on the "Pendulum-v0" at OpenAI Gym and showed oscillatory behavior as well. Despite the unstability of SVG(0), our approach shows significant improvement of the training performance versus the common approach (downsampled input of the all visual scene and undirected exploration). Namely, DDPG and SVG(0) solely could not get any progress at all under the delayed reinforce conditions. Please refer to the statistics in the Table 1 for further details.

Table 1: Experimental statistics

Parameter	SVG(0)+LSTM predictor	SVG(0)	DDPG
Runs	20	20	20
Successful runs	20	0	0
Episodes per run	350	500	500
Episodes to solve	$\sim$110	0	0

4 Conclusion

We have shown experimentally that our approach to unsupervised imitation learning (UIL) allows a RL agent to explore environments with delayed reinforce and large search space. Comparison of UIL and DDPG (driven by Ornstein-Uhlenbeck process [3]) and SVG(0) (under common settings [21]) revealed that efficient exploration is not achievable for models based on (un)directed exploration. Moreover, unsupervised imitation learning does not require to query or execute expert, and even does not require to designate an expert explicitly. That is why UIL improves performance significantly.

Furthermore, our approach to UIL extends and easily integrates into the previously investigated (un)directed exploration and supervised imitating learning. Namely, when behavioral correlations are weak, our approach tends to behave as undirected exploration. Moreover, it can be supplemented with heuristic methods such as directed exploration [1] and prioritized experience replay [7]. Fast [25] or One-shot [13] supervised imitation learning architectures could further underlie our approach to speed up the learning.

However, SVG(0) method was not an elegant choice of the framework for learning continuous control stochastic policies. SVG(0) was selected from its model-free perspective as well as similarity to DDPG for benchmarking purposes. SVG(0) turned out to be difficult to train and showed unstable and oscillatory behavior. Another source of instability was the random sampling of the experience. Prioritized experience replay could optimize taking traces from the replay memory. Moreover, the number of epochs were too small. We trained the RL agent for 70K samples, but according to [24, 3] the best result could be obtained with approximately 1M samples. Both these reasons as well as absence of immediate rewards explain the instability and why so many iterations were needed for such a simple task as the proposed under the toy environment.

We note that there is a large room for the improvement of our approach to unsupervised imitation learning.

Firstly, we consider raw pixel input processing. We assumed supervised classification of the raw pixel input by RCNN, that made our approach not completely unsupervised. The class attribute transfer [26] could be used for unsupervised classification. Partition of mask into individual masks for an object component [17] could improve the detection of moving objects orientation. Further, the recurrent feedback could improve classification in the case of occlusion. In addition, adding the agent motion information (orientation, speed) could allow to predict the visual scene changes caused by motion.

Secondly, the LSTM predictor efficiency could be improved with learnable spatial constants for pooling of the hidden states, Multi-hypothesis tracking [27]. STM combined with a Kalman filter [28] improves upon the original gradient descent learning algorithm making LSTM achieve even faster convergence and better performance. Moreover, we could apply recurrent batch normalization [29] to speed up learning and dropout [30] to prevent overfitting.

Thirdly, the stochastic policy was approximated by a forward fully-connected network. The policy network could have recurrent connections [31, 32] or LSTM [33] to learn new tasks in just a few trials.

References

[1] Sebastian B. Thrun. The role of exploration in learning control, 1992.

[2] Sebastian B. Thrun. Efficient exploration in reinforcement learning. Technical report, Carnegie Mellon University, 1992.

[3] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

[4] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1928–1937. JMLR.org, 2016.

[5] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. In Lee et al. [34], pages 4026–4034.

[6] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Rémi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. *CoRR*, abs/1611.01224, 2016.

[7] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *CoRR*, abs/1511.05952, 2015.

[8] Jürgen Schmidhuber. Adaptive confidence and adaptive curiosity. Technical report, Institut fur Informatik, Technische Universitat Munchen, Arcisstr. 21, 800 Munchen 2, 1991.

[9] Sebastian Thrun and Knut Möller. Active exploration in dynamic environments. In John E. Moody, Stephen Jose Hanson, and Richard Lippmann, editors, *Advances in Neural Information Processing Systems 4, [NIPS Conference, Denver, Colorado, USA, December 2-5, 1991]*, pages 531–538. Morgan Kaufmann, 1991.

[10] Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In Bruce W. Porter and Raymond J. Mooney, editors, *Machine Learning, Proceedings of the Seventh International Conference on Machine Learning, Austin, Texas, USA, June 21-23, 1990*, pages 216–224. Morgan Kaufmann, 1990.

[11] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In Yee Whye Teh and D. Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, volume 9 of *JMLR Proceedings*, pages 661–668. JMLR.org, 2010.

[12] Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, volume 15 of *JMLR Proceedings*, pages 627–635. JMLR.org, 2011.

[13] Yan Duan, Marcin Andrychowicz, Bradly C. Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. *CoRR*, abs/1703.07326, 2017.

[14] Denis Pogosov. Position-trajectory control of advanced tracked robots with diesel-electric powertrain. In Jong-Hwan Kim, Eric T. Matson, Hyun Myung, Peter Xu, and Fakhri Karray, editors, *Robot Intelligence Technology and Applications 2 - Results from the 2nd International Conference on Robot Intelligence Technology and Applications, RiTA 2013, Denver, Colorado, USA, December 18-20, 2013*, volume 274 of *Advances in Intelligent Systems and Computing*, pages 393–403. Springer, 2013.

[15] Stanley P. Franklin and Max H. Garzon. On stability and solvability (or, when does a neural network solve a problem?). *Minds and Machines*, 2(1):71–83, 1992.

[16] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. Visual SLAM algorithms: a survey from 2010 to 2016. *IPSJ Trans. Computer Vision and Applications*, 9:16, 2017.

[17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[18] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(6):1137–1149, 2017.

[19] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Fei-Fei Li, and Silvio Savarese. Social LSTM: human trajectory prediction in crowded spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 961–971. IEEE Computer Society, 2016.

[20] Qiaozhe Li, Xin Zhao, and Kaiqi Huang. Learning temporally correlated representations using lstms for visual tracking. In *2016 IEEE International Conference on Image Processing, ICIP 2016, Phoenix, AZ, USA, September 25-28, 2016*, pages 1614–1618. IEEE, 2016.

[21] Nicolas Heess, Gregory Wayne, David Silver, Timothy P. Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2944–2952, 2015.

[22] Yemi Okesanjo and Victor Kofia. Revisiting stochastic off-policy action-value gradients. *CoRR*, abs/1703.02102, 2017.

[23] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[24] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[25] Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. No-regret reductions for imitation learning and structured prediction. *CoRR*, abs/1011.0686, 2010.

[26] Christoph H. Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 951–958. IEEE Computer Society, 2009.

[27] Haifeng Gong, Jack Sim, Maxim Likhachev, and Jianbo Shi. Multi-hypothesis motion planning for visual object tracking. In Dimitris N. Metaxas, Long Quan, Alberto Sanfeliu, and Luc J. Van Gool, editors, *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*, pages 619–626. IEEE Computer Society, 2011.

[28] Juan Antonio Pérez-Ortiz, Felix A. Gers, Douglas Eck, and Jürgen Schmidhuber. Kalman filters improve LSTM network performance in problems unsolvable by traditional recurrent nets. *Neural Networks*, 16(2):241–250, 2003.

[29] Tim Cooijmans, Nicolas Ballas, César Laurent, and Aaron C. Courville. Recurrent batch normalization. *CoRR*, abs/1603.09025, 2016.

[30] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In Lee et al. [34], pages 1019–1027.

[31] Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl$ˆ2$: Fast reinforcement learning via slow reinforcement learning. *CoRR*, abs/1611.02779, 2016.

[32] Xiujun Li, Lihong Li, Jianfeng Gao, Xiaodong He, Jianshu Chen, Li Deng, and Ji He. Recurrent reinforcement learning: A hybrid approach. *CoRR*, abs/1509.03044, 2015.

[33] Nicolas Heess, Jonathan J. Hunt, Timothy P. Lillicrap, and David Silver. Memory-based control with recurrent neural networks. *CoRR*, abs/1512.04455, 2015.

[34] Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors. *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 2016.