AUTOMATIC CURRICULUM GENERATION VIA TASK PERTURBATIONS IN REINFORCEMENT LEARNING

Srinivas Venkattaramanujam, Riashat Islam & Doina Precup
School of Computer Science
McGill University
Montreal, H3A 0G4, Canada
{sri.venkattaramanujam@mail,riashat.islam@mail,dprecup@cs}.mcgill.ca

ABSTRACT

Automatic curriculum generation involves generating a sequence of tasks without requiring expert demonstrations or any form of reward shaping. Since hand designing reward functions are difficult in practice, curriculum learning alleviates this problem by generating multiple subtasks, often defined by goals and a simpler reward function. Most practical approaches to curriculum learning often involves training a separate model to generate goals or subtasks. We propose an approach that requires no external models to generate a curriculum in a continuous task space. We propose a simple technique for curriculum generation by random task perturbations in continuous task parameterization. We demonstrate this in the context of goal-conditioned policies and show that our approach is on par with a method that uses a generative model to generate new goals.

1 INTRODUCTION

Reinforcement learning algorithms has had several recent successes, ranging from solving the game of Go (Silver et al., 2016) to efficiently solving Atari games mnih2015humanlevel to robotic control and manipulation DBLP:journals/corr/LillicrapHPHETS15. In most of these domains, the inherent assumption is that the goal of the agent is to maximize the cumulative return objective based on a single reward function. However, a long lasting goal of AI is for agents to simultaneously solve multiple tasks and be able to transfer domain knowledge from one task to another without additional computational complexity. Solving multiple tasks in a multi-task learning setup often poses challenges due to the requirement of re-defining the objective function to maximize several objectives.

Curriculum learning has been proposed to efficiently solve multiple tasks sequentially. It involves having a predefined (Bengio et al., 2009) or a learned (Graves et al., 2017) schedule to generate a set of related multiple tasks, to break down a difficult problem into easier smaller problems. Recently an automatic curriculum generation procedure for Reinforcement Learning agents has been proposed in Florensa et al. (2018) using goal conditioned policies (Kaelbling, 1993; Schaul et al., 2015b). Often goal conditioned methods allow for tasks to be specified using a collection of goals and simpler the reward functions characterized by the goals. Therefore, goal conditioned methods are suited for tasks in which specifying an appropriate reward function is tedious. As such, curriculum learning also helps in sparse reward challenging exploration problems where the set of generated related tasks can guide the agent towards an effective exploration strategy. Often, a good exploration strategy relies on identifying or discovering the task structure, and be able to exploit the task structure. We focus in the context of multi-task learning where the agent can solve multiple tasks at the same time, through the perspective of automatic curriculum learning. We propose that an efficient way to solve multiple tasks also relies on good exploration strategies as the agent should identify useful substructures within the environment, where each sub-task has its own reward function. Towards this goal, we propose a surprisingly simple approach towards efficient curriculum generation in a multi-task learning context that results in efficient exploration using goal conditioned policies where the agent can generate its own reward functions through goals and solve these subtasks which naturally leads to efficient exploration of the environment.

Unlike previous works (Florensa et al. (2018), Florensa et al. (2017)) which requires additional complexity, we propose a simple approach of random perturbations of automatically generated goals to generate new goals. We show that by using goal conditioned policies and reward functions specific to a given goal, a natural way towards exploration is to randomly perturb the goals and generate new goals. This leads to random reward functions being generated through perturbations, and the goal of the agent is to solve all the generated goals in a multi-task learning setup. Our work is similar to previous work on automatic goal generation setups as in Florensa et al. (2018), but instead of using a generative model such as GAN (Goodfellow et al., 2014) to generate goals, we hypothesize that a simple curriculum generation strategy is to generate new goals by randomly perturbing current goals. In our experiments, following a similar setup as in Florensa et al. (2018), we show that our simple strategy can work quite effectively in complex maze continuous control tasks.

2 PRELIMINARIES

We consider the reinforcement learning setup where the goal of the agent is to maximize cumulative returns $J(\pi) = \sum_{t=0}^{\infty} \gamma^t r_t$. In this work, we consider goal conditioned policies $\pi(a|s, g)$ which are often introduced to provide additional information to policies, as in Universal Value Functions (UVFs) and several recent works considering latent or goal conditioned policies (Goyal et al., 2019). In a multi-task learning objective, often the goal space defines a subtask that is required for the agent to solve, where the goals are either part of the state space or generated by a trained generative model (Nair et al., 2018; Florensa et al., 2018), including VAEs (Kingma & Welling, 2014) or GANs. Several work has also recently looked at curriculum learning objectives, or in a student teacher framework where the goal of the teacher is to provide useful training signals for the student to solve multiple tasks.

Automatic Goal Generation using GAN : We propose that goals generated by perturbing the current goals provide useful signals for the agent to increasingly improve by solving multiple tasks defined by the reward functions. We detail our approach in the next section, which is based on Florensa et al. (2018) and so we briefly review it here. The reward function r^g, parameterized by the goal g, is an indicator function where $r^g(s) = 1$ if $s = g$ and 0 $otherwise$. The advantage of using indicator reward function is that no prior knowledge about the task is required other than being able to determine whether the goal has been reached or not. In continuous goal spaces the probability of exactly reaching any particular goal is 0 and hence r^g in this case is defined as $r^g(s) = 1$ if $d(s, g) < \epsilon$, where d is a distance metric in the goal space and ϵ is a threshold. The indicator reward function enables the interpretation of the undiscounted return, $R^g = \sum_{t=1}^{T} r_t^g$, as a $bernoulli$ random variable indicating whether the policy has achieved its task of reaching the specified goal.

The overall objective is to find a policy $\pi^*(a|(s, g))$ that is optimal for all goals $g \in G$ drawn according to a test distribution $D(g)$ i.e find a policy π^* $s.t$ $\pi^* = \arg\max_\pi E_{g \sim D_g}[R^g(\pi)]$. The test distribution $D(g)$ is assumed to be uniform over G. Training the agent by sampling the goals uniformly from D_g leads to significantly slower learning since initially the majority of the goals will be beyond the capability of the agent and hence returns are sparse leading to slower progress as shown in Florensa et al. (2018). Therefore it is desirable to have a curriculum that progressively increases the difficulty of the goals. In order to push the frontier of the agent's capabilities it is necessary to train on goals that have not been previously learned to achieve but such goals must not be too difficult for the current ability of the agent, which will otherwise result in poor training signal. In order to balance this trade-off, a notion of Goals of Intermediate Difficulty *(GOID)* is introduced in Florensa et al. (2018) as explained below.

Goal Labeling and Goal Generation : $GOID$ goals are defined as the goals which are reached with probability $\in [success_{min}, success_{max}]$ where $0 < success_{min} < success_{max} < 1$ are the minimum and maximum probabilities of successfully achieving the goal under the current policy respectively. New *GOID* goals are generated by sampling from a GAN trained to generate the *GOID* goals. The goals are then labeled as easy, *GOID* and difficult goals when the probability of reaching the goal is $[0, success_{min})$, $[success_{min}, success_{max}]$ and $(success_{max}, 1]$ respectively. Two approaches to compute the probability of reaching a goal given in Florensa et al. (2018) are as follows: (i) For each goal, perform multiple rollouts following the current policy to compute the probability of achieving the goal. These trajectories are not used in policy optimization and (ii) for each goal, use the trajectories used in the previous iteration of policy optimization to compute the

probability of achieving the goal. In order to avoid catastrophic forgetting, a fixed proportion of easy goals are used along with the *GOID* goals in the policy optimization phase. After the goal labeling phase, a GAN is trained using the *GOID* goals and new goals are generated by sampling from the GAN.

3 PROPOSED METHOD

Our objective is to train an agent in a curriculum learning setting, without introducing additional complexity in generating a set of tasks. We propose to use goal conditioned policies in the multi-task learning setup, with the objective to maximize cumulative returns along the set of tasks. Several works have proposed using goal conditioned policies or value functions, either as an intrinsic reward or curiosity based objective Burda et al. (2018), or in a multi-task learning setup (Goyal et al., 2019). We extend the recent work from Florensa et al. (2018) to use goal conditioned policies in the curriculum learning setup. In most recent works, the way to generate the goals has been to use generative models to generate goals (Florensa et al., 2018) (Nair et al., 2018).We develop a simple approach based on the intuition of random perturbations to generate new set of tasks, where each task is rewarding for the agent to solve in a curriculum learning setup. We hypothesize that the following are key properties to generating goals for multi-task learning, and demonstrate that the following desired properties can be achieved using a non-stationary buffer to store the goals in tandem with goal perturbation. A detailed description of the algorithm and non-stationary replay buffer is provided in the next section.

Non-Stationarity: We require a non-stationary process to generate goals that are suited to the agent's current capabilities, and ensure not to generate goals that has already been learned by the agent. This is crucial to ensure that the generated goals aid in expanding the agent's capability.
Local Interpolation: In order to generate goals that are of similar difficulty to *GOID* goals and generalize to nearby goals, it is necessary to have a local interpolation of the goals but also avoid global interpolation such as interpolation between modes since such goals might be infeasible and therefore result in wasted training capacity.
Stability: To avoid the sub-optimal solution of focusing on only a part of the goal space that is more likely under the current policy and avoid specialization in only that regions of the goal space that are likely under the current policy, we need to ensure that the new goals generation must enable exploration in all the regions of the goal space previously visited. To achieve this end, the goal generation process must be insensitive changes in policy.

3.1 GOAL GENERATION USING A GOAL BUFFER

To generate goals using a buffer, we employ 2 separate buffers to store easy goals and to store $GOID$ goals respectively. The agent is trained using a fixed proportion of the easy goals and *GOID* goals in order to avoid catastrophic forgetting while attempting to expand its abilities. *GOID* goals buffer is used to store goals that are of intermediate difficulty and also to generate new goals by applying perturbations to the sampled goals. The implementation of the easy goals buffer is an ordinary list but the $GOID$ buffer is designed to satisfy all of the properties desired from a generative models using the following techniques:

Non-stationarity and Stability: Maintaining a balance between non-stationarity and stability is a trade-off that we have to balance. It is a trade-off because having a infinitely long buffer provides maximum stability but not non-stationarity whereas a a buffer of size 1 provides maximum non-stationarity but no stability. In order to ensure non-stationarity, it is necessary to discard the goals that were *GOID* goals several iterations ago since such goals would have been learned to be solved. The simplest mechanism to discard old goals is to implement a fixed size queue i.e use a first-in first-out policy to discard the goals from the buffer. However, without any restrictions on the number of goals inserted, the entire buffer can get replaced if the number of new goals is greater the maximum capacity of the queue and therefore cause instability. Hence, it is necessary to restrict the number of new goals that can be inserted into the buffer in each iteration in order to ensure stability.

Local interpolation: An efficient curriculum is one in which the difficult of the tasks presented to the agent are increased progressively and are therefore are near the agent's current capabilities. Therefore, in order to progressively increase the difficulty of the goals, it is desirable to generate

new goals that are close to the $GOID$ goals. In the case of a continuous parameterization of the goal space, this can be achieved by adding noise to the *GOID* goals. In order to achieve this, we have made the sampling from the replay buffer to be noisy i.e a few goals are sampled from the buffer and a small perturbation is applied to the sampled goals to generate new goals. Therefore, our *GOID* buffer is a queue of fixed size with the restriction of only updating a fraction of the goals in each iteration and the new *GOID* goal generation is achieved by simply sampling from the $GOID$ buffer followed by the addition of a small noise. Adding noise in this manner, like using a generative model, may result in the generation of infeasible states but those states are subsequently filtered out due the labeling phase.

3.2 Policy Optimization

For training $\pi(a|s, g)$, we uniformly sample a fixed proportion of easy and *GOID* goals from the buffers every K iterations and then sample a goal among the chosen goals for a given episode. We use a policy optimization based objective, such as TRPO Schulman et al. (2015) and after K steps of policy optimization, the goals are labeled as easy, hard or *GOID* goals. The easy goals and the *GOID* goals are stored in their respective buffers and the hard goals are discarded. The labeling procedure helps eliminate the unfeasible goals generated by applying of noise to the *GOID* goals and also eliminate goals that are reliably solved so that the boundaries of the agent's capabilities are progessively pushed. Our training loop is shown below:

procedure TRAIN **Input:** π_0 **Output:** π_{final}
  $trajectories \leftarrow$ trajectories generated according to π_0
  $GOID\ goals\ buffer \leftarrow$ Goals derived from the $trajectories$ ▷ {transform states to goals}

  $easy\ goals\ buffer \leftarrow \Phi$ ▷ {empty list}
for $i \leftarrow 1$ *to* N **do**
  $sample\ goid\ goals \leftarrow$ sample from goid goals buffer ▷ {as shown in Algorithm 2}
  $sample\ easy\ goals \leftarrow$ sample from easy goals buffer
  $all\ goals \leftarrow$ concatenate(sample goid goals, sample easy goals)
  $\pi_i \leftarrow$ train policy(π_{i-1}, $all\ goals$) ▷ {Perform K iterations of policy optimization by conditioning the policy in each episode on a goal sampled uniformly from $all\ goals$ }
  $labels \leftarrow$ label each goal in all goals as $easy$, $hard$ and $GOID$ goals
  $goid\ goals \leftarrow$ select $GOID$ goals from all goals using labels
  $easy\ goals \leftarrow$ select easy goals from all goals using labels
  $goid\ goals\ buffer \leftarrow$ update goid goals buffer (goid goals) ▷ {as shown in Algorithm 1}
  $easy\ goals\ buffer \leftarrow$ update easy goals buffer (easy goals)

end

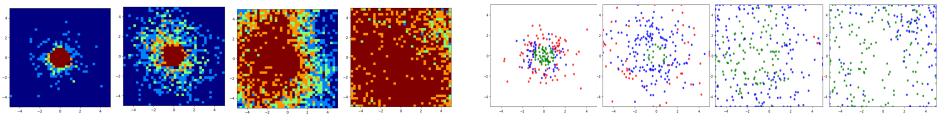
4 Experimental Results

We evaluate our proposed method for automatic generation of curriculum via task perturbations, on a range of continuous control MuJoCo (Todorov et al., 2012) tasks . We follow a similar experimental procedure as Florensa et al. (2018), where in two of the environments, the agent is an ant with 41 dimensional state space placed in an open or U-shaped 2D maze in a $[-5, 5]^2$ grid. In the third experiment, we use a point-mass agent with 4 dimensional state space placed in an enclosed 2D maze as shown in Fig 4. A detailed description of the environments can be found in Florensa et al. (2018) and Duan et al. (2016). The goal space is the (x, y) coordinates denoting the position of the center of mass of the agent in the maze. The objective in all the tasks is to be able to reach any specified goal in the goal space and the performance of the agent is measured by the proportion of the maze area reachable by agent, known as the coverage of the maze, when conditioned on goals from those regions. In all these environments, the goal is considered reached when the COM of the agent is within an ϵ_r-ball of the goal.

We highlight that the key benefit of our approach is that we do not need to train a separate generative model for task generation, and experimentally demonstrate that very similar performance can in fact be achieved by perturbation of the goals as show in Figure 3. The only case where there is a

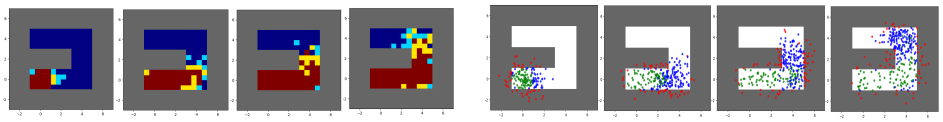
notable difference between using a GAN and task perturbation is in the Maze Ant environment using separate rollouts to label the goals. Since, using separate rollouts to label the goals is ineffective, it is unlikely the case the to be used in practice. Figures 1 and 2 demonstrates the non-stationarity of the *GOID* buffer. The green goals near the the starting positions are due to the goals from the easy goals buffer. It can be seen that the *GOID* goals generated from the *GOID* buffer progressively moves away from the starting region, resulting in pushing the agent's frontier. The goals generated by local perturbations are robust to the case of multimodal goal distribution as shown in Figure 4 *(d)*.

In order to show that stability of the *GOID* buffer is crucial and being sensitive to changes in the policy will lead to a sub-optimal performance, we experiment with the case of generating goals that is dependent entirely on the current policy. This is done by generating new goals from the trajectories visited by the policy by sampling the states visited by the policy during the optimization phase in the previous iteration. We show that in the Free Ant environment, generating goals in this manner results in sub-optimal exploration of the maze as show in Figure 5. Further experimental results are given in Appendix.



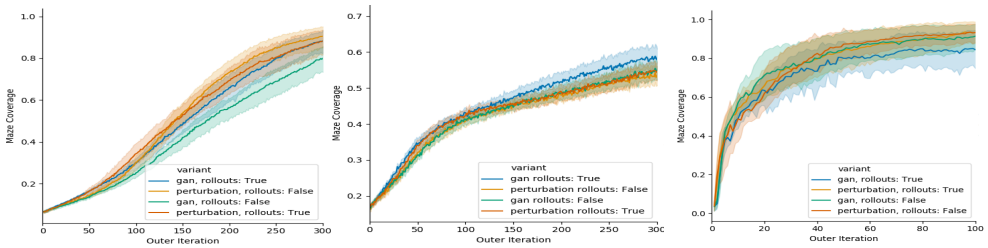
(a) Itr 1 (b) Itr 50 (c) Itr 150 (d) Itr 250 (e) Itr 1 (f) Itr 50 (g) Itr 150 (h) Itr 250

Figure 1: *Free Ant: (a) - (d)* shows the evolution of the coverage of the maze. The agent reaches the goals in the brown region with probability 1 and never achieves goals in the blue region. *(e) - (h)* shows the evolution of the new goals generated for the agent along with the goals from easy goals buffer. Green indicates easy goals, Blue indicates *GOID* goals and red indicates difficult goals



(a) It1 1 (b) Itr 50 (c) Itr 150 (d) Itr 300 (e) Itr 1 (f) Itr 50 (g) Itr 150 (h) Itr 300

Figure 2: *Maze Ant:* Evolution of the coverage and goals provided to the agent



(a) Free Ant (b) Maze Ant (c) Multipath Point-Mass

Figure 3: Coverage Plots: All the experiments are averaged over 5 random seeds. *gan* indicates that the goals were generated using GoalGAN, *perturbation* indicates that goals were generated by perturbing the goals, *rollouts:True* indicates that goals are labeled performing 4 rollouts for each goal and *rollouts:False* indicates that goals are labeled using the trajectories used in policy optimization.

5 Conclusion and Future work

In this work, we propose a simple approach towards curriculum learning, based on random task perturbations in continuous task parameterization. We demonstrate that instead of learning a generative model to generate new related tasks in curriculum, we can instead add random perturbations to existing tasks to sequentially generate a series of related tasks. Our experiments demonstrate that this approach can work equally well, or in some cases, better than using generative models in curriculum learning. In future, we aim to study methods to generate continuous representation of tasks and evaluate whether such representation is amenable to curriculum generation through perturbations.

REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *CoRR*, abs/1707.01495, 2017. URL http://arxiv.org/abs/1707.01495.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 41–48, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-516-1. doi: 10.1145/1553374.1553380. URL http://doi.acm.org/10.1145/1553374.1553380.

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. *CoRR*, abs/1604.06778, 2016. URL http://arxiv.org/abs/1604.06778.

Carlos Florensa, David Held, Markus Wulfmeier, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. *CoRR*, abs/1707.05300, 2017. URL http://arxiv.org/abs/1707.05300.

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1515–1528, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/florensa18a.html.

Vincent François-Lavet, Yoshua Bengio, Doina Precup, and Joelle Pineau. Combined reinforcement learning via abstract representations. *CoRR*, abs/1809.04506, 2018. URL http://arxiv.org/abs/1809.04506.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pp. 2672–2680, Cambridge, MA, USA, 2014. MIT Press. URL http://dl.acm.org/citation.cfm?id=2969033.2969125.

Anirudh Goyal, Riashat Islam, DJ Strouse, Zafarali Ahmed, Hugo Larochelle, Matthew Botvinick, Sergey Levine, and Yoshua Bengio. Transfer and exploration via the information bottleneck. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJg8yhAqKm.

Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. *CoRR*, abs/1704.03003, 2017. URL http://arxiv.org/abs/1704.03003.

Leslie Pack Kaelbling. Learning to achieve goals. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, Chambery, France, 1993. Morgan Kaufmann. URL http://people.csail.mit.edu/lpk/papers/ijcai93.ps.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6114.

Ashvin Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *CoRR*, abs/1807.04742, 2018. URL http://arxiv.org/abs/1807.04742.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pp. 1312–1320, 2015a. URL http://jmlr.org/proceedings/papers/v37/schaul15.html.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1312–1320, Lille, France, 07–09 Jul 2015b. PMLR. URL http://proceedings.mlr.press/v37/schaul15.html.

Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proceedings of the First International Conference on Simulation of Adaptive Behavior on From Animals to Animats*, pp. 222–227, Cambridge, MA, USA, 1990. MIT Press. ISBN 0-262-63138-5. URL http://dl.acm.org/citation.cfm?id=116517.116542.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015. URL http://arxiv.org/abs/1502.05477.

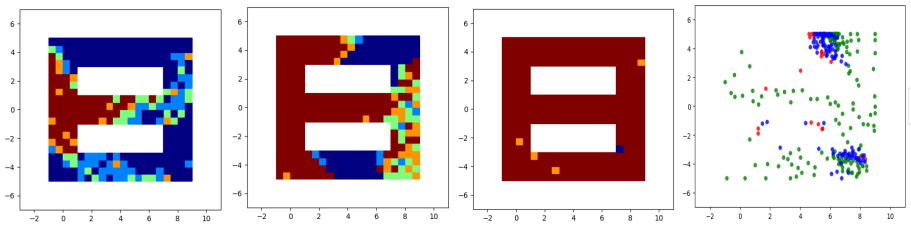
David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, jan 2016. ISSN 0028-0836. doi: 10.1038/nature16961.

Sainbayar Sukhbaatar, Ilya Kostrikov, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *CoRR*, abs/1703.05407, 2017. URL http://arxiv.org/abs/1703.05407.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, pp. 5026–5033. IEEE, 2012. ISBN 978-1-4673-1737-5. URL http://dblp.uni-trier.de/db/conf/iros/iros2012.html#TodorovET12.

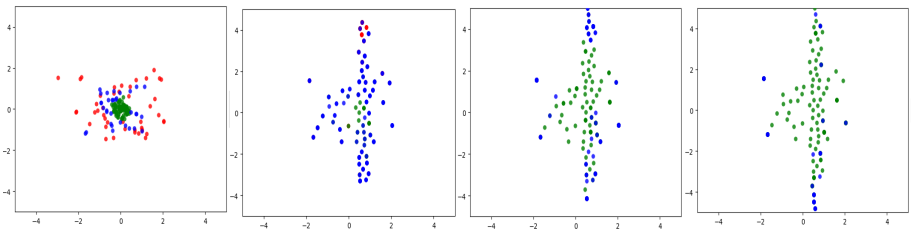
A ADDITIONAL EXPERIMENTAL DETAILS

We use TRPO with GAE as the policy optimization algorithm and train the agent for 500 outer iterations (corresponding to N in 3.2). In each outer iteration, the policy optimization phase consists of 5 inner iterations and in each inner iteration $100,000$ samples and $20,000$ samples for the ant and multi-path point-mass agents respectively are collected to perform policy optimization. The maximum horizon of each episode is 500 and 400 for ant and point-mass agents respectively. In each outer iteration, we sample 100 easy goals and 200 *GOID* goals. We bootstrap our method by generating the initial goals by performing rollouts according to a random policy and subsample 200 goals from the collected random goals and label them. Using the labels we fill the easy and *GOID* goals buffer with the respective goals. The architecture used for policy optimization is a 2 layer neural network with 32 hidden units and *tanh* non-linearity and the discount factor is 0.998 and $GAE\lambda = 0.995$. The terminal ϵ for ant and point-mass agents are 1 and 0.3 respectively and the noise added to perturb the goals is gaussian noise of 0 mean and 0.5 variance. The maximum buffer size is 300 and the number of goals added in each iteration is 30.



(a) Iteration 1 (b) Iteration 10 (c) Iteration 50 (d) Multimodal *GOID* goals

Figure 4: Evolution of the goals in Multi-path point-mass: Green indicates easy goals, Blue indicates *GOID* goals and red indicates difficult goals



(a) Iteration 1 (b) Iteration 50 (c) Iteration 150 (d) Iteration 250

Figure 5: Suboptimal coverage by generating goals from the trajectories

Environment	**GoalGAN (rollout)**	**GoalGAN**	**Perturbation (rollout)**	**Perturbation**
Maze Ant	0.59	0.55	0.55	0.54
Free Ant	0.88	0.8	0.88	0.91
Multipath Point Mass	0.85	0.91	0.93	0.93

Table 1: Summary of the results

B *GOID* GOAL BUFFER UPDATE

We highlight that the when goals are inserted to the goal buffers, only goals that are at least ϵ_r away from goals present in the buffer are inserted. There are two benefits of using such an approach: a) reduced memory requirement and b) it can interpreted as making the goal distribution within the

buffer to be uniform since any information about the distribution of the goals to be inserted are discarded due to the filtering by distance.

Algorithm 1

Algorithm 1: Update $GOID$ buffer

global variables
$GOID\ goals\ buffer$ ▷ {stored $GOID\ goals$}
$MAX\ size$ ▷ {maximum size of $GOID\ goals\ buffer$}
$\#max\ new\ goals$ ▷ {maximum number of $goals$ appended in one iteration}
end global variables

procedure UPDATE
Input: $GOID$ goals, ϵ_r ▷ {ϵ_r is the minimum distance between two goals in the buffer}
# inserted goals $\leftarrow 0$
for *for each goal in* $GOID$ *goals* **do**
  if *distance(goal, GOID goals buffer)* $\geq \epsilon_r$ **then**
    insert $goal$ to $GOID\ goals\ buffer$
    #inserted goals $\leftarrow$ #inserted goals + 1
    if # *inserted goals* $\geq \#max\ new\ goals$ **then**
      exit loop
    end
  end
end
if *size(GOID goals buffer)* $>$ *MAX Size* **then**
  $selectedgoals \leftarrow$ Uniformly sample $size(GOID\ goals\ buffer) - MAX\ Size$ goals
  discard $selected\ goals$
end

C GENERATE NEW *GOID* GOALS

Algorithm 2

Algorithm 2: Sample from $GOID$ buffer

global variables
$GOID\ goals\ buffer$ ▷ {stored $GOID\ goals$}
end global variables

procedure SAMPLE
Input: n, $noise$ ▷ {noise of each component}
Output: $GOID\ goals$
$selected\ goals \leftarrow$ uniformly sample n goals from $GOID\ goals\ buffer$
$selected\ goals \leftarrow$ selected goals + $noise$
return $selected\ goals$

D RELATED WORK

The need for conditioning policies on the goal information has been pointed in several works, where the goal information can be used to identify useful subgoals or exploiting the task structure of the environment. UVFAs Schaul et al. (2015b) has been proposed previously where conditioning the value functions on the goal information has also been shown to be extremely useful to share the semantics of the value function across several tasks, and several works built up from that, including HER (Andrychowicz et al., 2017). Often goal conditioned policies were also introduced from the information bottleneck perspective, looking at the mutual information between states and goals to

identify useful subgoals and provide this as a form of intrinsic motivation or exploration bonus for efficient transfer and exploration (Goyal et al., 2019).

Goal conditioned policies have also been proposed previously as a form of intrinsic motivation Schaul et al. (2015a) or as a measure of curiosity Burda et al. (2018). Recent approaches include the discovery of bottleneck states (Goyal et al., 2019) or learning a feature space (François-Lavet et al., 2018). Exploration can also be formulated of as an agents internal drive towards learning more about the environment. This is often defined as intrinsic motivation, or curiosity of the agent (Pathak et al., 2017; Schmidhuber, 1990). Curiosity or intrinsic motivation can therefore be thought of as a task agnostic exploration heuristic towards the goal of learning in an online fashion based on the agents interactions with the environment.

Exploration can also be seen from the perspective of automatic curriculum generation. By being able to generate subtasks or subgoals for the agent to solve, by using generative models such as GANs as in Florensa et al. (2018), the agent can also discover efficient exploration strategies in a multi-task learning setup. Curriculum learning has gained recent interest, due to the ability to solve multiple tasks at the same time, and being able to use prior knowledge from these tasks for transfer. This has also be explored in the framework of self-play (Sukhbaatar et al., 2017), where the agent generates objectives to itself (as a form of intrinsic motivation) and tries to outperform these objectives in a progressive manner. This can also be viewed as the agent providing itself a form of exploration bonus to solve the single reward function tasks faster.