Continual Reinforcement Learning deployed in Real-life using Policy Distillation and Sim2Real Transfer

Anonymous Authors[1]

Abstract

We focus on the problem of teaching a robot to solve tasks presented sequentially, i.e., in a continual learning scenario. The robot should be able to solve all tasks it has encountered, without forgetting past tasks. We provide preliminary work on applying Reinforcement Learning to such setting, on navigation tasks for a 3 wheel omni-directional robot. Our approach takes advantage of state representation learning and policy distillation. Policies are trained using learned features as input, rather than raw observations, allowing for better sample efficiency. Policy distillation is used to combine different policies into a single policy that solves all encountered tasks.

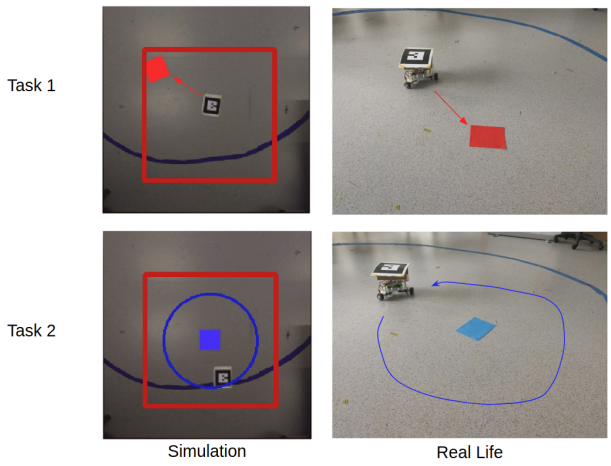

Figure 1. Having access to task 1 only first, and then task 2 only, we learn a single policy that solves two real-life navigation tasks using policy distillation and sim2real transfer.

1. Introduction

In realistic real-life reinforcement learning scenarios, for example involving service robots, tasks evolve over time, either because the context of one task changes or because new tasks appear (Doncieux et al., 2018). Our end goal is therefore to have an embodied agent in real-life that learns incrementally as time passes. One example would be a robot tasked with wrapping gifts. Most gifts are rectangular packages (cuboids), so the robot would first learn to wrap cuboids. Then if a soccer ball appears, the robot has to learn how to wrap a sphere while still being able to wrap cuboids after that. The robot should add this knowledge to his past knowledge. Even if it would be easier to learn to wrap spheres and cuboids before test time, there are potentially many other shapes that have to be considered, and thus, learning continually seems more natural and convenient than trying to learn all at once.

Continual Learning (CL) and State Representation Learning (SRL) are essential to build agents that face such challenge. SRL allows to build strong representation of the world since agents should be able to understand their surroundings, and extract general concepts from sensory inputs of complex scenes. An agent better sees a chair as an object, not as a bunch of pixels together in an image. CL allows to learn such representation without forgetting in settings where the distribution of data change through time and is needed for agents that learn in the real-world and are required to adapt to changes. Combining CL and SRL would then allow to create strong representation robust to catastrophic forgetting.

Reinforcement learning (RL) is a popular approach to learn robot controllers that also has to face the CL challenges, and can take advantage of SRL to learn faster and to produce more robust policies. Therefore we perform our experiments (Fig. 1) in a setup where tasks are encountered sequentially and not all at once. Note that it differs from a setting where we can pick and shuffle experiences, often encountered in the multi-task RL literature (cf section 2.2).

In our approach we aim to take advantage of simulations to create this scenario. We demonstrate that deploying a policy in real-life, which has continually learned two tasks in simulation is successful with our approach.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Our contribution consists on applying two major paradigms for robotics in real life: a state representation learning approach for compact and efficient representation that facilitates learning a policy, and a policy that learns continually in a sequential manner. The approach is deployed in a real robot thanks to policy distillation and sim2real transfer. Furthermore in opposition to most method in reinforcement learning, at test time, the solution we propose does not need a task indicator. Indeed, the information about the task to solve can be found in the image.

The rest of the article is structured as follows. Section 2 introduces state representation learning, multi-task RL and continual learning paradigms in an RL setting, Sec.4 details the robotics settings and tasks performed; and Sec.3 details the methods utilized and Sec.6 concludes with future insights from our experiments.

2. Related work

2.1. State representation learning (SRL)

Scaling end-to-end reinforcement learning to control real robots from vision presents a series of challenges, in particular in terms of sample efficiency. Against end-to-end learning, SRL (Lesort et al., 2018) can help learn a compact, efficient and relevant representation of states. Previous works as (Finn et al., 2015; Watter et al., 2015; van Hoof et al., 2016; Lesort et al., 2017; Sermanet et al., 2017; Thomas et al., 2017; Raffin et al., 2019) has shown that SRL can speed up policy learning, reducing the number of samples needed while additionally being easier to interpret.

2.2. Multi-task RL

Multi-task RL aims at constructing one single policy module that can achieve a number of different tasks. The CURIOUS algorithm (Colas et al., 2018) selects through exploration, the tasks to be learned that improve an absolute learning progress metric the most. Policy distillation (Rusu et al., 2015) can also be used to merge different policies into one module/network. The Distral algorithm (Teh et al., 2017) is one successful example of such approach: a shared policy distills common behaviours from task-specific policies. Then, the distilled policy is used to guide task-specific policies via regularization using a Kullback-Leibler (KL) divergence. Other approaches like SAC-X (Riedmiller et al., 2018) or HER (Andrychowicz et al., 2017) take advantage of Multi-task RL by learning auxiliary tasks in order to help the learning of an objective task.

2.3. Continual Learning

Continual learning (CL) is the ability of a model to learn new skills without forgetting previous knowledge. In our context, it means learning several tasks sequentially and being able to solve any task at the end of the sequence.This differs from the easier multi-task scenario, where tasks can be experienced all at once.

Most CL approaches can be classified into four main methods that can also be used for classification and generation. Those four methods differ from the way they handle memory of past tasks. The first method, referred to as *rehearsal*, keeps samples from previous tasks (Rebuffi et al., 2017; Nguyen et al., 2017). The second method, *regularization*, either by constraining weight updates in order to maintain knowledge from previous tasks (Kirkpatrick et al., 2017; Zenke et al., 2017; Maltoni & Lomonaco, 2018) or keeping old model as memory and distilling knowledge (Hinton et al., 2015) later to remember (Li & Hoiem, 2018; Schwarz et al., 2018; Rusu et al., 2015) The third category of strategy, *dynamic network architectures*, maintains past knowledge thanks to architecture odification while learning (Rusu et al., 2016; Fernando et al., 2017; Li & Hoiem, 2018; Fernando et al., 2017). The fourth and more recent method is *generative replay* (Shin et al., 2017; Lesort et al., 2018; Wu et al., 2018), where a generative model is used as a memory to produce samples from previous tasks. This approach has also been referred to as pseudo-rehearsal.

2.4. RL in real-life

Applying RL to real-life scenarios is a major challenge that has been studied widely. Most attempts fall into two categories: games and robotics.

For games, AlphaGo Zero (Silver et al., 2018) has mastered the game of Go from scratch without any human supervision by combining RL, self-play and Monte Carlo Tree Search (Chaslot et al., 2008). AlphaStar (Vinyals et al., 2019) and OpenAI Five (OpenAI, 2018) were both able to get competitive results against professional human players on the game Starcraft and DOTA2, respectively. Both solutions are based on RL, and current research is still investigating how to master the game with the same constraints as humans (e.g. same FPS).

In robotics, there is a plethora of successful attempts at deploying RL on real robots. One common approach is training policies in simulation and then deploying them in real-life hoping that they will successfully transfer, considering the gap in complexity between simulation and the real world. Such approaches are termed *Sim2Real*(Golemo, 2018), and have been successfully applied (Christiano et al., 2016; Matas et al., 2018) in many scenarios. In order to cope with the unpredictable nature of the real world, one can use Domain Randomization (Tobin et al., 2017), which we use in our approach. This technique trains policies in numerous simulations that are randomly different from each other (different backgroud colors, etc.). That way, the transfer to

real life is easier.

Others have tried to train policy directly on real robots, facing the hurdle of the lack of sample efficiency that RL suffers from. SAC-X (Riedmiller et al., 2018) is one example where a successful policy is learned directly on the real robot.

In the literature, most approaches focus on the single-task or simultaneous multi-task scenario. In this paper, we attempt to train a policy on several tasks sequentially and deploy it in real life. Hence, we attempt to apply RL in real life in a continual learning setting.

3. Methods

In this section we present the method proposed to combine state representation learning (SRL) and continual learning (CL) in a real life reinforcement learning setting. First we present how a single task is learned and how does the SRL part works, secondly we explain how to learn continually and thirdly we explain how we evaluate learning in the different phase of the learning sequence.

3.1. Learning on one task

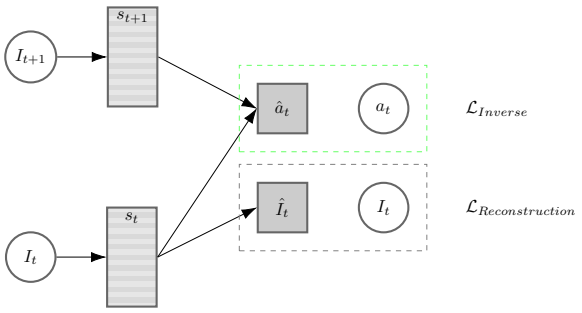

Figure 2. SRL Combination model: combines a reconstruction of an image I prediction and an inverse dynamic models losses in the state representation s. Arrows represent inference, dashed frames represent losses computation, rectangles are state representations, circles are real observed data, and squares are model predictions.

Each task is learned according to the same procedure we describe here. First, as we use a SRL approach, we need to learn a state representation encoder. We sample data from the environment Env_t with an agent guided by a random policy. We call this dataset D_{Rt}. D_{Rt} is then used to train a SRL model composed of an inverse model and an auto-encoder. This architecture is inspired from (Raffin et al., 2019), and illustrated in Fig.2.

Once the SRL model is trained, we only keep the encoder E_t and use it to learn our policy Π_t on top of it using reinforcement learning with the model $M(\theta)$ (θ represent the model parameters). Once Π_t is learned, we use it to generate sequences of on-policy data with associated action, which will eventually be used for distillation (Fig. 3, left). We call this distillation dataset $D_{\Pi t}$. We generate $D_{\Pi t}$ in the following way : we sample randomly a starting position and then let the agent generate a trajectory. At each step we save observation and associated action. We stop the sequence when enough reward is gathered (see section 4).

From each task is only kept the dataset $D_{\Pi t}$. As soon as we change task, D_{Rt} and Env_t are not available anymore.

In our setting, in order to decrease training time, we generate D_{Rt} in simulation and learn Π_t also in simulation. However, in the end of T tasks, $\Pi_{D0,\dots,T-1}$ is tested in a real robot. In order, to pass the reality gap, the datasets generated are augmented with luminosity variation.

3.2. Learning continually

To learn continually we use a distillation method (Rusu et al., 2015). Once we learned several tasks, we can aggregate several distillation datasets and distill the knowledge into a new model $M_Dt(\theta')$ to produce a single monolithic policy (Fig. 3, right). θ' are the parameters of the distillation model.

The distillation consists in learning in a supervised fashion the action probability associated to an image. Each dataset $D_{\Pi t}$ allows to distill the policy Π_t into a new network. We name the distilled policy Π_{Dt}. With the aggregation of several distillation datasets, we can distill several policies into the same network. By extension to the previous denomination, a model where policy 1 and policy 2 have been distilled in, is called $\Pi_{D1,2}$.

At test time, we do not need a task indicators, but assumes that the observations and state space allows to recognize the current task. In the context of continual RL, the task signal is mandatory if the observation does not give any clue about the policy to run. In our setting, as the policy can be inferred, we do need it necessary.

The method presented allows to learn continually several policies without forgetting. On the other hand, $M(\theta)$ also learn on the sequence of task but without any memorization mechanism, its leads to catastrophic forgetting. The dataset $D_{\Pi t}$ contains 10k samples per tasks which allows to learn the distillation very quickly (a few minutes are needed to learn Π_{Dt} when several hours are needed to learn Π_t).

3.3. Evaluation

The main evaluation is the performance of the final single policy, which can supposedly achieve all previous tasks, as well as being deployed in real life. For that, we report the mean and standard error on 5 runs of the policy on each task in simulation 4, and provide videos to show the behaviour

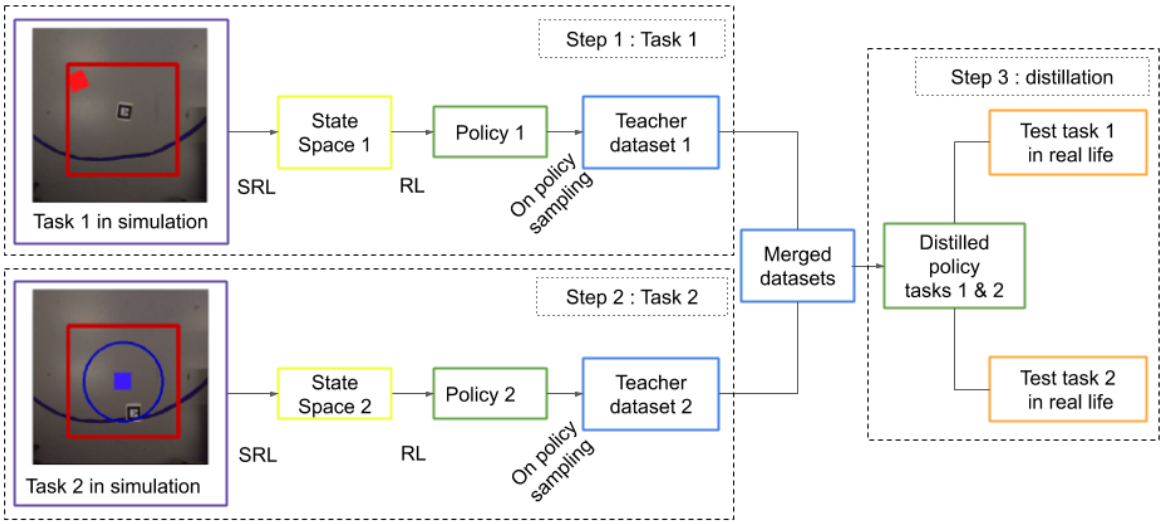

Figure 3. Summary of the experimental setup. Step 1 and 2 correspond to learning policies for task 1 and 2, and using those to create distillation datasets. Step 3 is the distillation of the two policies into a single policy which can be deployed in simulation and on the real robot.

of the final policy.

In an other hand we also would like to analyze the learning process. In order to have an insight of the evolution of the distilled model, we save distillation datasets at different checkpoints in the sequence of task. Those checkpoint are saved regularly during the RL training.

By distilling and evaluating at several time step, we are capable to evaluate the evolution of learning and forgetting on all environments separately and jointly. At each checkpoint, we evaluate the actual policy Π_t on past tasks to evaluate forgetting and compare it to $\Pi_D 0, .., t$.

It is important to note that, even if we consider Env_t as not available anymore at $t+1$, we did use it for evaluation purpose at any time.

4. Experimental setup

We apply our approach to learn continually two navigation tasks on a real mobile robot.

4.1. Robotic setup

The experiments consists of navigation tasks using a 3 wheel omni-directional robot. It is similar to the 2D random target mobile navigation ((Raffin et al., 2018), identical reward setting and possibility of movement). The robot is identified by a black QR code and the scene is recorded from above.

We are able to simulate the experiment, since the robot's input is a fixed RGB image of the scene recorded from above. The robot uses 4 high level actions (move left/right, move up/down in a cartesian plane relative to the robot) rather than motor commands.

The room where the real-life robotic experiments are to be performed is subject to illumination changes. The input image is a top-down view of the floor, which is lighted by surroundings windows and artificial illumination of the room. Hence, the illumination changes depending on the weather and time of the day. We use domain randomization (Tobin et al., 2017) to improve the chances of the policies learned in simulation to better transfer to the real world, by being robust to the weather and time of the day. During RL training, at each timestep, the color of the background is randomly changed.

4.2. Continual learning setup

Our continual learning scenario is composed of two similar environments, where the robot is asked to do different tasks. In environment 1, the robot is asked to reach a red square marker (task 1). In environment 2, it is asked to circle around a blue square marker (task 2).

It is important to note that as the target for different task are from different color, the algorithms can automatically infer which policy it need to run and then does not need task label at test time.

While generating $D_\Pi t$ (see section 3.1) we stop the sequence when 8 rewards are accumulated in a row for the task 1 (it means that the robot successfully reach the red target and stayed on it) and for task 2 we stop after 250 time step (so the robot have time to circle around the blue target).

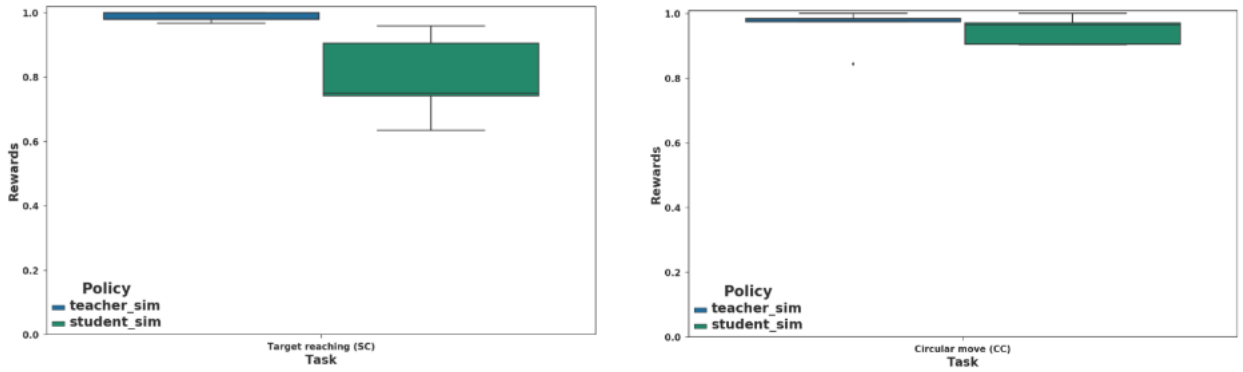

Figure 4. Comparison between performance (normalized mean reward and standard error) of policy trained on one task only to distilled student policy on the two tasks. The student policy has similar performance on both tasks. **Left:** *Reach target* task. **Right:** *Circle around* task

5. Results

5.1. Main result

Our main result is the continual learning of a single policy that solves both tasks in simulation, as presented in Fig.3 [1]. The two teacher policies are learnt separately on each environment, sequentially. Then, distillation is used to combine the two teacher policies into a single policy that can solve the two tasks.

Fig.4 demonstrate the efficiency of our approach. We can see that the single student distilled policy achieve close to maximum reward in both tasks.

5.2. Evaluation of distillation

We performed a more explicit evaluation of distillation in the task 2 (circular movement). While we train a policy using RL, we save the policy every 200 episodes (50000 timesteps), and distill it into a new student policy which we test. This is illustrated in Fig.5. Both curves are very close, which indicates distillation works as intended. It is able to transfer a policy using only a limited distillation dataset, with limited loss in the policy performance.

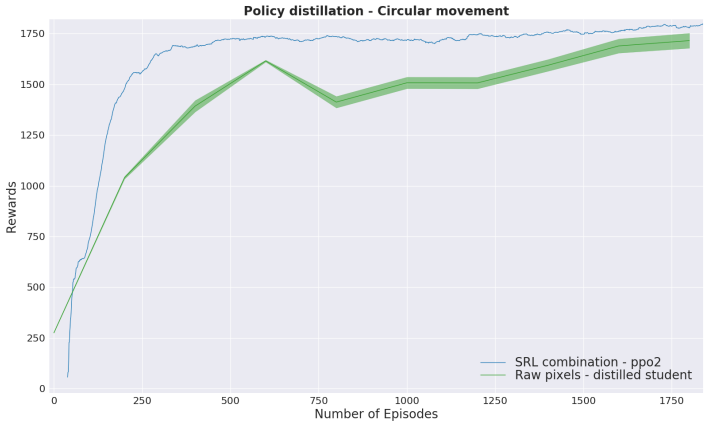

Figure 5. Demonstration of the effectiveness of distillation. Blue: RL training curve of PPO2 on task circular. Green: Mean and std performance on 8 seeds of distilled student policy. At each point, the blue policy is used to be distilled in a student policy. Both curves are very close, which indicates distillation works as intended.

6. Discussion and future work

Our work is preliminary and offers many possibilities for improvement. Our roadmap include having not only a policy learned in a continual way, but also the SRL model associated. We would need to update the SRL model as new tasks are presented sequentially. One possible approach would be to use Continual SRL methods like S-TRIGGER (Caselles-Dupré et al., 2019) or VASE (Achille et al., 2018).

[1]The deployment and evaluation in real life is part of future work

We also expect to encounter issues when scaling continual learning approaches to more tasks or environments. Indeed, the agent should not accumulate knowledge blindly, but rather make connections between different types of information (i.e. generalize) and/or selectively forget non-useful knowledge.

Moreover, we intend to soon provide with supplementary quantitative results and videos of these tasks deployed in the real-life setup.

We would like to train policies directly on the real robot, as it is the end goal scenario for this research. One promising approach would be to use Model-Based RL on models learning during the SRL phase to improve sample efficiency to the point at which training on a real robot takes a reasonable amount of time.

7. Conclusion

In this paper we provide preliminary results towards a proper real life continual learning setup, where a real robot would encounter tasks presented in a sequence and be asked to accumulate knowledge in a scalable manner. The building blocks for achieving a single policy that solves all presented tasks are RL using state representation models and distillation into a single policy which is a good candidate for transfer to real life.

References

Achille, A., Eccles, T., Matthey, L., Burgess, C., Watters, N., Lerchner, A., and Higgins, I. Life-long disentangled representation learning with cross-domain latent homologies. In *Advances in Neural Information Processing Systems*, pp. 9873–9883, 2018.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, O. P., and Zaremba, W. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pp. 5048–5058, 2017.

Caselles-Dupré, H., Garcia-Ortiz, M., and Filliat, D. S-trigger: Continual state representation learning via self-triggered generative replay. *arXiv preprint arXiv:1902.09434*, 2019.

Chaslot, G., Bakkes, S., Szita, I., and Spronck, P. Monte-carlo tree search: A new framework for game ai. 2008.

Christiano, P., Shah, Z., Mordatch, I., Schneider, J., Blackwell, T., Tobin, J., Abbeel, P., and Zaremba, W. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*, 2016.

Colas, C., Sigaud, O., and Oudeyer, P.-Y. Curious: Intrinsically motivated multi-task, multi-goal reinforcement learning. *arXiv preprint arXiv:1810.06284*, 2018.

Doncieux, S., Filliat, D., Díaz-Rodríguez, N., Hospedales, T., Duro, R., Coninx, A., Roijers, D. M., Girard, B., Perrin, N., and Sigaud, O. Open-ended learning: a conceptual framework based on representational redescription. *Frontiers in Neurorobotics*, 2018.

Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A. A., Pritzel, A., and Wierstra, D. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

Finn, C., Tan, X. Y., Duan, Y., Darrell, T., Levine, S., and Abbeel, P. Deep spatial autoencoders for visuomotor learning. *CoRR*, abs/1509.06113, 2015. URL http://arxiv.org/abs/1509.06113.

Golemo, F. *How to Train Your Robot - New Environments for Robotic Training and New Methods for Transferring Policies from the Simulator to the Real Robot*. Theses, Université de Bordeaux, December 2018. URL https://hal.inria.fr/tel-01974203.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Lesort, T., Seurin, M., Li, X., Díaz-Rodríguez, N., and Filliat, D. Unsupervised state representation learning with robotic priors: a robustness benchmark. *CoRR*, abs/1709.05185, 2017. URL http://arxiv.org/abs/1709.05185.

Lesort, T., Caselles-Dupré, H., Garcia- Ortiz, M., Stoian, A., and Filliat, D. Generative Models from the perspective of Continual Learning. *arXiv e-prints*, art. arXiv:1812.09111, December 2018.

Lesort, T., Díaz-Rodríguez, N., Goudou, J.-F., and Filliat, D. State representation learning for control: An overview. *Neural Networks*, 2018.

Li, Z. and Hoiem, D. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2018.

Maltoni, D. and Lomonaco, V. Continuous learning in single-incremental-task scenarios. *arXiv preprint arXiv:1806.08568*, 2018.

Matas, J., James, S., and Davison, A. J. Sim-to-real reinforcement learning for deformable object manipulation. *arXiv preprint arXiv:1806.07851*, 2018.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

OpenAI. Openai five. https://blog.openai.com/openai-five/, 2018.

Raffin, A., Hill, A., Traoré, R., Lesort, T., Díaz-Rodríguez, N., and Filliat, D. S-rl toolbox: Environments, datasets and evaluation metrics for state representation learning. *arXiv preprint arXiv:1809.09369*, 2018.

Raffin, A., Hill, A., Traoré, K. R., Lesort, T., Díaz-Rodríguez, N., and Filliat, D. Decoupling feature extraction from policy learning: assessing benefits of state representation learning in goal based robotics. *Workshop on Structure and Priors in Reinforcement Learning (SPiRL) at ICLR*, 2019.

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

Riedmiller, M., Hafner, R., Lampe, T., Neunert, M., Degrave, J., Van de Wiele, T., Mnih, V., Heess, N., and Springenberg, J. T. Learning by playing-solving sparse reward tasks from scratch. *arXiv preprint arXiv:1802.10567*, 2018.

Rusu, A. A., Colmenarejo, S. G., Gulcehre, C., Desjardins, G., Kirkpatrick, J., Pascanu, R., Mnih, V., Kavukcuoglu, K., and Hadsell, R. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Schwarz, J., Luketina, J., Czarnecki, W. M., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., and Hadsell, R. Progress & compress: A scalable framework for continual learning. *arXiv preprint arXiv:1805.06370*, 2018.

Sermanet, P., Lynch, C., Hsu, J., and Levine, S. Time-contrastive networks: Self-supervised learning from multi-view observation. *CoRR*, abs/1704.06888, 2017. URL http://arxiv.org/abs/1704.06888.

Shin, H., Lee, J. K., Kim, J., and Kim, J. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pp. 2990–2999, 2017.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Teh, Y., Bapst, V., Czarnecki, W. M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., and Pascanu, R. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 4496–4506, 2017.

Thomas, V., Pondard, J., Bengio, E., Sarfati, M., Beaudoin, P., Meurs, M., Pineau, J., Precup, D., and Bengio, Y. Independently controllable factors. *CoRR*, abs/1708.01289, 2017. URL http://arxiv.org/abs/1708.01289.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30. IEEE, 2017.

van Hoof, H., Chen, N., Karl, M., van der Smagt, P., and Peters, J. Stable reinforcement learning with autoencoders for tactile and visual data. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3928–3934, Oct 2016. doi: 10.1109/IROS.2016.7759578.

Vinyals, O., Babuschkin, I., and Chung, J. *AlphaStar blog post*, 2019. URL https://bit.ly/2B5YrKh.

Watter, M., Springenberg, J., Boedecker, J., and Riedmiller, M. Embed to control: A locally linear latent dynamics model for control from raw images. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28*, pp. 2746–2754. Curran Associates, Inc., 2015.

Wu, C., Herranz, L., Liu, X., Wang, Y., van de Weijer, J., and Raducanu, B. Memory replay gans: learning to generate images from new categories without forgetting. *arXiv preprint arXiv:1809.02058*, 2018.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3987–3995, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/zenke17a.html.