State of the Art on: Generative models for reinforcement learning

Pierluca D'Oro, pierluca.doro@gmail.com

1. Introduction to the research topic

Machine learning (ML) is the computational field that leverages statistical techniques to develop algorithms able to learn from data. *Reinforcement learning* (RL) [59] is a subfield of ML that deals with the problem of training an agent to maximize a reward signal while acting in an environment. The study of *generative models* is another ML subarea, with the aim of describing in probabilistic terms the process that generated the data. Despite several families of algorithms can be applied to RL and generative modeling, in recent years *deep learning* approaches have demonstrated great capabilities for some problems: they aim at learning distributed, highly compositional representations of data through end-to-end gradient-based optimization.

Relevant conferences for this research topic, according to experts' opinion and rankings, are AAAI and IJCAI, in the broader field of artificial intelligence, NIPS and ICML, specifically for machine learning, and ICLR, specialized in deep learning; relevant journals are instead the *Journal of Machine Learning Research* (Microtome), *Transactions on Pattern Analysis and Machine Intelligence* (IEEE), *Machine Learning* (Springer), *Transactions on Neural Networks and Learning Systems* (IEEE).

1.1. Preliminaries

Reinforcement Learning

Reinforcement learning is the study of sequential decision-making problems in which an agent acts in an environment with the aim of maximizing a cumulative reward. The problem can be formalized as a *Markov decision process* (MDP) [50], which, in the broader case of a stochastic environment, is defined by a state space $\mathcal{S}$, an action space $\mathcal{A}$, an initial state distribution $p_0(s_0)$, a state transition distribution $\mathcal{T}(s_t|s_{t-1}, a_{t-1})$, a reward function $r : S \times A \to \mathbb{R}$, a discount factor $\gamma \in [0, 1]$. A *trajectory* is a sequence $\tau_{t_0:T} = s_{t_0}, a_{t_0}, r_{t_0}, ..., s_T, a_T, r_T$ resulting from the agent iterating through the process of observing a state, performing an action, and receiving a reward. Trajectories occur according to a probability distribution $p(\tau)$. In an MDP, the Markov property holds: transitions to a state depend only on the previous state and action, or, more formally, $\mathcal{T}(s_t|s_{t-1}, a_{t-1}) = \mathcal{T}(s_t|s_0, a_0, ..., s_{t-1}, a_{t-1})$ for any trajectory. The agent decides how to act according to a *policy*, a probability distribution $\pi(a_t|s_t)$ over the actions. In the case of environments that are not fully observable, the agent has only access to an observations o_t of the actual state, and it will act according to a policy $\pi(a_t|o_t)$. In RL, we usually want to estimate the optimal policy π^* by maximizing the cumulative expected discounted reward $r(\tau_{t_0:T}) = \mathbb{E}[\sum_{t=t_0}^{T} \gamma^{t-t_0} r(s_t, a_t)]$ the agent obtains during its sequential interactions with the environment. A low value of γ penalizes rewards that are later in time.

Generative models

Generative models are used to perform *density estimation* of the probability distribution $p(\mathbf{x})$ from which some data are drawn. An estimate $\hat{p}(\mathbf{x})$ can be obtained either *explicitly*, by defining a parameterized $\hat{p}(\mathbf{x}; \theta)$, or *implicitly*, usually being able to sample from $\hat{p}(\mathbf{x})$. A common approach is to maximize the log-likelihood of the data, or, equivalently, to minimize $D_{KL}(p(\mathbf{x})||\hat{p}(\mathbf{x}))$, the *Kullback–Leibler divergence* between the real and the estimated distribution. Nonetheless, this log-likelihood is not tractable and simplifying assumptions are needed.

There are three main categories of likelihood-based methods that explicitly learn $\hat{p}(\mathbf{x})$. *Autoregressive approaches* use *fully observed* models, without any latent variables to explain the hidden factors of the data. They try to make

the problem tractable using a factorization $\hat{p}(\mathbf{x}) = \prod_{t=0}^{T} p(x_t|x_0, ..., x_{t-1})$, corresponding to the generation, for a sample, of a single feature at a time [26, 62, 61]. *Flow methods* leverage invertible transformations on random variables [16, 17, 37]. *Variational methods* [38, 52] try to learn an *approximate density* by maximizing a computable $\mathcal{L}(\mathbf{x}; \theta)$ subject to the *evidence lower bound* $\mathcal{L}(\mathbf{x}; \theta) \leq \log \hat{p}(\mathbf{x}; \theta)$; a common approach is to train a *variational autoencoder* (VAE), using $-\mathcal{L}(\mathbf{x}; \theta)$ as a loss function. A VAE is composed of an *encoder* that maps the input data into a latent representation and a *decoder*, able both to reconstruct the encoded data and to generate new samples, decoding points sampled from the prior distribution of the latent variables.

Generative adversarial networks (GANs) are *implicit density models* that provide a way to sample from $\hat{p}(\mathbf{x})$: in GAN training, we look for a *Nash equilibrium* in the game between a generator G, producing samples resembling the real ones, and a discriminator D, which distinguishes the samples drawn from the dataset from the ones produced by G. Several value functions were devised for this game, each of which leads to an equivalent minimization of a divergence between $p(\mathbf{x})$ and $\hat{p}(\mathbf{x})$ (e.g., Jensen-Shannon [25], Wasserstein [3], Pearson χ^2 [41]). Refer to [24] for a more precise taxonomy of recent generative models.

Tools

Current research on machine learning often takes the form of open source software projects. *Python*, a general-purpose interpreted programming language, is currently the most used in the field. Several libraries were developed for its scientific use, many of which are based on the *scientific Python* (SciPy) ecosystem. Moreover, in the field of *deep learning*, large-scale software libraries for creating and training models on GPUs exist. These usually feature *automatic differentiation* and other utilities for gradient-based optimization, and offer the possibility to compose and use parameterized functional modules into complex structures at runtime: this concept is sometimes referred to as *differentiable programming*. Two popular projects, started by industry research groups, are *Pytorch* [48] and *Tensorflow* [1]. In reinforcement learning, tools such as the *Arcade Learning Environment* [7], *OpenAI gym* [9], *DeepMind Lab* [6] and *rllab* [18] provide a common interface and simulation environment for learning agents.

1.2. Research topic

The key ingredient of any reinforcement learning algorithm is the experience collected by the agent. Recent reinforcement learning techniques have been able to achieve superhuman results [43, 55] for problems in which, despite the extreme difficulty of the task, data collection occurs easily through the use of simulation. Nonetheless, in almost all real world scenarios, the interaction between agent and environment, required to collect experience for learning, entails high costs. Namely, if we compare this interaction with the one an agent can have in a simulated environment, it is usually significantly *slower*, *more expensive*, and *more dangerous*. For instance, if you want to instruct an agent to perform autonomous driving, you may think to train it in the real world. However, this needs huge amounts of on-the-road hours, fees and expenses, together with particular attention to avoid accidents.

Generative models can help reinforcement learning methods to face this issue in multiple ways. They are often used to approximate the dynamics of the environment: this fast, cheap and safe estimate can be then used by an RL algorithm to limit the interaction required with the actual environment. The reinforcement learning methods of this kind are called *model-based*, as opposed to *model-free* approaches that do not use any explicit approximation of environment dynamics. Other ways to leverage generative modeling in addressing the problems mentioned above are related to the reuse of already-acquired knowledge, once again to reduce the number of interactions with the environment. In fact, it is desirable that an agent is able either to share knowledge across tasks, known as *transfer*, or to be resilient to shifts in state distribution, known as *domain adaptation*. Estimating the generating distribution of the data can be the foundation for methods that favor both these capabilities.

2. Main related works

2.1. Classification of the main related works

Model-based RL is an established field with recognized roots [58, 44, 67]. Many RL approaches use a generative model to learn an estimate of the transition distribution $\mathcal{T}(s_t|s_{t-1}, a_{t-1})$. Some of them try to overcome the *model bias*, due to the imperfect estimation of the dynamics, considering uncertainty with techniques based on Gaussian processes [15] or Bayesian neural networks [23, 36]. Other work has leveraged the progress in recurrent neural networks [33] to relax the Markov assumption and take longer sequences into account in generating subsequent states. In some cases [19, 47], the generative modeling of the environment has been directly done in *sensory space* (e.g., the *pixel space* for visual tasks), while in others, more efficiently, the latent representations of the raw observations have been considered [13, 10, 63, 66].

The representation of observations is often obtained by means of convolution-based variational autoencoders [29, 20]. In addition, variational methods have also been used to facilitate the transfer across tasks [31] and goals [46]. Recently, they have been employed in estimating $p(\tau)$, the probability distribution of whole trajectories [42, 14], since unrolling the one-step models of the dynamics autoregressively several times generates poor predictions, practically inhibiting long-horizon planning. Nonetheless, this approach has not yet been fully explored and deserves more attention. Some methods combine *model-based* and *model-free* approaches, with the objective of making the latter more data efficient and adaptable. Although it has been a well-known integration for a long time [58], it can fully exploit the representation capabilities of modern deep learning models [51, 27, 45]. Other approaches have used predictive models of the environment to encourage exploration [53, 60, 49, 57, 11, 2, 54].

Generative adversarial networks have shown potential in RL for tree-based planning [4], adapting observations across domains [8], generating goals [21] or trajectories to reach them [39], efficiently learning from demonstrations [32, 65, 40, 5, 22]. However, recent advances in generative modeling using GANs [35, 69, 64, 68], mainly achieved within the computer vision community, have enormous potential left for application in reinforcement learning.

Recent mentioned works on the use of generative models for reinforcement learning are shown in Table 1.

2.2. Brief description of the main related works

Many recent works analyzed how to properly design and leverage different forms of *recurrent environment models* for the benefit of learning agents. Preliminary results in [47] showed that it is possible to directly condition a one-step video prediction model using actions, obtaining an estimate of the behavior of a visual environment. The video prediction model is a deep encoder-decoder network that uses convolutional and recurrent layers for modeling visual and temporal aspects respectively. This approach is improved in [13], where a latent encoding is predicted in place of the whole high-dimensional state and alternative training schemes are considered to achieve better autoregressive unrolling. In [10], environment simulators capable of stochastic estimates are evaluated for the same problem considering also the possibility for an agent to actively query the predictor.

Several RL approaches jointly learn a representation of observations from the environment and policies (for *policy search* approaches) or value functions (for *value-based* approaches). The obtained representation is extremely unhandy for knowledge transfer: on the one hand, it prevents reusing the same policy if the distribution of the observations changes; on the other hand, it prevents reusing the same state representation if the task changes. To address this problem, it is beneficial to learn a latent embedding of observations and a policy separately. [29] proposes a modular method composed of a VAE-based vision module, a predictive MDN-RNN [28] model and a decision-making component. The three modules are learned one after the other and the simple linear controller takes into account both the current observation representation and the history encoded by the model of the environment. The approach also exploits the learned Doom video game's dynamics to train a policy in the hallucinated environment it obtains by unrolling the generative model and employing it in the actual game. In [31], a β-VAE [30] is used for learning a *disentangled* representation of images, whose peculiar features are therefore represented independently. This allows a policy trained on a source task to be capable of *zero-shot* domain adaptation, acting directly on a target task without further retraining. The method proposed in [46]

for goal-oriented reinforcement learning exploits a VAE as well, for multiple purposes: first, to obtain a latent representation to be used for learning a policy and a value function in a model-free setting; secondly, to obtain the goal-dependent reward, defined as $r(s, g) = -||e(s) - e(g)||_2$, the opposite of the Euclidean distance between the representations computed by the VAE encoding function e of a generic state s and a goal state g; third, to generate new goals for learning by sampling from the VAE prior.

The capabilities of generative models have been extensively leveraged for planning. [63] and [19] use two different video prediction models to estimate the visual dynamics of the environment and determine the best action to be taken according to *model predictive control* (MPC), an optimal control technique that plans in a finite horizon but recompute the action to take after each time step. An MPC is also used in [66], which exploits a VAE to linearize the local dynamics of image trajectories. In [51], Monte Carlo rollouts of environment simulators are interpreted by the agent to determine its policy, in a model-based/model-free hybrid. GANs have been used for planning as well: in [39] a type of generative adversarial networks named InfoGAN [12] is used to obtain a state representation, subsequently used to plan; in [4], a common GAN-based method for image-to-image translation [34] is employed to estimate state transitions and, together with a reward estimator, to use Monte Carlo tree search for playing Atari games.

Generative adversarial networks have been also employed in *imitation learning*, the task of learning policies from expert demonstrations. The problem is commonly framed as in [32], considering the generator as the policy to be learned and training the discriminator to distinguish between expert and generated actions. Many approaches have built upon [32], for instance by improving performance on multimodal trajectory distributions [65] or by including differentiation through estimated environment dynamics [5].

Recently, some approaches have tried to model the multi-step dynamics of the environment with a single generative step, using an approximation $\hat{p}(\tau)$ of the trajectory distribution. In [42], a conditional VAE [56], based on dilated causal convolution [61], is conditioned on past states and actions as well as on planned future actions and trained on *temporal segments* to be able to sample likely future trajectories. The method is then improved with the use of an additional *latent action prior* obtained by means of another VAE to ensure that future actions can only be sampled according to a feasible distribution given past actions. The evaluation is done on control in two settings: *trajectory optimization*, maximizing rewards obtained over the predicted trajectories, and *policy optimization*, in which a policy acting on temporal segments is learned. [14] exploits variational inference to embed state trajectories, employing the learned model for performing hierarchical reinforcement learning. A lower level policy, which acts as an additional decoder for the VAE, is constrained to be consistent with the actual *state decoder*, and it is steered by a *model predictive controller* acting at the trajectory latent-space level.

2.3. Discussion

Transferring skills among compatible tasks and similar environments is a crucial problem in reinforcement learning. Although it has been studied for several years, it is still far from being solved. Humans are able to learn how to perform a task and easily acquire the ability to perform similar tasks; they can quickly understand when and how to use previously learned skills to solve a new problem. All agents trained by nowadays algorithms are instead intelligent in a much *narrower* sense. Being able to effectively reuse knowledge is a fundamental ingredient for widening their intelligence.

Model-based reinforcement learning, obtained through the use of generative models, is a promising route to improve RL algorithms. Although the real world is complex and difficult to model, even just the use of a good estimate of its dynamics can rule out all the behaviors of an agent that are not consistent with it. In this way, both the transferability and the sample-efficiency can be improved. Moreover, progress in model-based RL can yield fruitful combinations of machine learning, optimal control and classical artificial intelligence techniques.

An open issue in the use of generative models for reinforcement learning concerns the kind of distribution these models are called to approximate. The vast majority of existing approaches estimated single transitions sampled from $\hat{\mathcal{T}}(s_t|s_{t-1}, a_{t-1})$. However, unrolling this kind of predictive models for a large number of time steps leads to progressively more inaccurate estimates, due to both compounding estimation errors and environment

uncertainty. Therefore, modeling $p(\tau)$ for whole trajectories can be an effective strategy. However, it has not yet been explored enough, nor integrated with previously proposed one-step methods.

Generative Models for Reinforcement Learning											
Paper	**LfD**	**Plan.**	**Tran.**	**Goal**	**Eff.**	**Expl.**	$\hat{p}(\tau)$	$\hat{\mathcal{T}}$	**VAE**	**GAN**	**F.O.**
Deisenroth and Rasmussen [2011]					✓			✓			✓
Wahlström et al. [2015]		✓			✓			✓			✓
Stadie et al. [2015]						✓		✓			✓
Watter et al. [2015]		✓			✓			✓	✓		
Gal et al. [2016]					✓			✓			✓
Gu et al. [2016]		✓			✓	✓		✓			✓
Finn et al. [2016]					✓			✓			✓
Ho and Ermon [2016]	✓									✓	
Agrawal et al. [2016]		✓		✓	✓			✓			✓
Wang et al. [2017]	✓		✓							✓	
Baram et al. [2017]	✓							✓		✓	
Finn and Levine [2017]		✓	✓					✓			✓
Li et al. [2017]	✓		✓							✓	
Mishra et al. [2017]		✓			✓		✓		✓		
Racanière et al. [2017]	✓		✓		✓			✓			✓
Killian et al. [2017]			✓					✓			✓
Higgins et al. [2017b]			✓						✓		
Florensa et al. [2017]				✓							✓
Pathak et al. [2017]						✓		✓			✓
Buesing et al. [2018]					✓			✓			✓
Shyam et al. [2018]		✓				✓		✓			✓
Nair et al. [2018]				✓					✓		
Ha and Schmidhuber [2018]					✓			✓	✓		✓
Co-Reyes et al. [2018]		✓			✓	✓	✓		✓		
Kurutach et al. [2018]		✓			✓			✓		✓	
Nagabandi et al. [2018]		✓			✓			✓			✓
Bousmalis et al. [2018]			✓		✓					✓	
Fu et al. [2018]	✓		✓							✓	

Table 1: Summary of recent works mentioned in this document on the application of generative models in reinforcement learning. The features used to describe them are, from left to right in the table header: the application for learning from demonstrations (LfD), planning, transfer, goal-based RL, sample efficiency, exploration; whether they model the trajectory distribution $p(\tau)$ or single-step transitions drawn from $\mathcal{T}$; the type of generative model that is employed, among VAE, GAN and fully observed (F.O.).

References

[1] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

[2] Agrawal, P., Nair, A. V., Abbeel, P., Malik, J., and Levine, S. (2016). Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems*, pages 5074–5082.

[3] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein Generative Adversarial Networks. In *International Conference on Machine Learning*, pages 214–223.

[4] Azizzadenesheli, K., Yang, B., Liu, W., Brunskill, E., Lipton, Z. C., and Anandkumar, A. (2018). Sample-efficient deep rl with generative adversarial tree search. *arXiv preprint arXiv:1806.05780*.

[5] Baram, N., Anschel, O., Caspi, I., and Mannor, S. (2017). End-to-end differentiable adversarial imitation learning. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 390–399, International Convention Centre, Sydney, Australia. PMLR.

[6] Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., Schrittwieser, J., Anderson, K., York, S., Cant, M., Cain, A., Bolton, A., Gaffney, S., King, H., Hassabis, D., Legg, S., and Petersen, S. (2016). DeepMind Lab. *arXiv:1612.03801 [cs]*.

[7] Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279.

[8] Bousmalis, K., Irpan, A., Wohlhart, P., Bai, Y., Kelcey, M., Kalakrishnan, M., Downs, L., Ibarz, J., Pastor, P., Konolige, K., et al. (2018). Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4243–4250. IEEE.

[9] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). OpenAI Gym. *arXiv:1606.01540 [cs]*.

[10] Buesing, L., Weber, T., Racaniere, S., Eslami, S., Rezende, D., Reichert, D. P., Viola, F., Besse, F., Gregor, K., Hassabis, D., et al. (2018). Learning and querying fast generative models for reinforcement learning. *arXiv preprint arXiv:1802.03006*.

[11] Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. (2018). Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*.

[12] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180.

[13] Chiappa, S., Racaniere, S., Wierstra, D., and Mohamed, S. (2017). Recurrent environment simulators. *arXiv preprint arXiv:1704.02254*.

[14] Co-Reyes, J. D., Liu, Y., Gupta, A., Eysenbach, B., Abbeel, P., and Levine, S. (2018). Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. *arXiv preprint arXiv:1806.02813*.

[15] Deisenroth, M. and Rasmussen, C. E. (2011). Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472.

[16] Dinh, L., Krueger, D., and Bengio, Y. (2014). Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*.

[17] Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016). Density estimation using Real NVP. *arXiv:1605.08803 [cs, stat]*.

[18] Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. (2016). Benchmarking Deep Reinforcement Learning for Continuous Control. *arXiv:1604.06778 [cs]*.

[19] Finn, C. and Levine, S. (2017). Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793.

[20] Finn, C., Tan, X. Y., Duan, Y., Darrell, T., Levine, S., and Abbeel, P. (2016). Deep spatial autoencoders for visuomotor learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 512–519. IEEE.

[21] Florensa, C., Held, D., Geng, X., and Abbeel, P. (2017). Automatic Goal Generation for Reinforcement Learning Agents. *arXiv:1705.06366 [cs]*.

[22] Fu, J., Luo, K., and Levine, S. (2018). Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*.

[23] Gal, Y., McAllister, R., and Rasmussen, C. E. (2016). Improving pilco with bayesian neural network dynamics models. In *Data-Efficient Machine Learning workshop, ICML*.

[24] Goodfellow, I. (2016). NIPS 2016 Tutorial: Generative Adversarial Networks. *arXiv:1701.00160 [cs]*.

[25] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

[26] Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

[27] Gu, S., Lillicrap, T., Sutskever, I., and Levine, S. (2016). Continuous Deep Q-Learning with Model-based Acceleration. In *International Conference on Machine Learning*, pages 2829–2838.

[28] Ha, D. and Eck, D. (2018). A neural representation of sketch drawings. In *International Conference on Learning Representations*.

[29] Ha, D. and Schmidhuber, J. (2018). Recurrent World Models Facilitate Policy Evolution. *arXiv:1809.01999 [cs, stat]*.

[30] Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017a). beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*.

[31] Higgins, I., Pal, A., Rusu, A. A., Matthey, L., Burgess, C. P., Pritzel, A., Botvinick, M., Blundell, C., and Lerchner, A. (2017b). Darla: Improving zero-shot transfer in reinforcement learning. *arXiv preprint arXiv:1707.08475*.

[32] Ho, J. and Ermon, S. (2016). Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573.

[33] Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780.

[34] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. *CVPR*.

[35] Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2018). Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*.

[36] Killian, T. W., Daulton, S., Konidaris, G., and Doshi-Velez, F. (2017). Robust and Efficient Transfer Learning with Hidden Parameter Markov Decision Processes. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 6250–6261. Curran Associates, Inc.

[37] Kingma, D. P. and Dhariwal, P. (2018). Glow: Generative Flow with Invertible 1x1 Convolutions. *arXiv:1807.03039 [cs, stat]*.

[38] Kingma, D. P. and Welling, M. (2013). Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*.

[39] Kurutach, T., Tamar, A., Yang, G., Russell, S., and Abbeel, P. (2018). Learning Plannable Representations with Causal InfoGAN. *arXiv:1807.09341 [cs, stat]*.

[40] Li, Y., Song, J., and Ermon, S. (2017). InfoGAIL: Interpretable Imitation Learning from Visual Demonstrations. *arXiv:1703.08840 [cs]*.

[41] Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., and Smolley, S. P. (2017). Least squares generative adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2813–2821. IEEE.

[42] Mishra, N., Abbeel, P., and Mordatch, I. (2017). Prediction and Control with Temporal Segment Models. In *International Conference on Machine Learning*, pages 2459–2468.

[43] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*.

[44] Moore, A. W. and Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13(1):103–130.

[45] Nagabandi, A., Kahn, G., Fearing, R. S., and Levine, S. (2018). Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE.

[46] Nair, A., Pong, V., Dalal, M., Bahl, S., Lin, S., and Levine, S. (2018). Visual Reinforcement Learning with Imagined Goals. *arXiv:1807.04742 [cs, stat]*.

[47] Oh, J., Guo, X., Lee, H., Lewis, R. L., and Singh, S. (2015). Action-conditional video prediction using deep networks in atari games. In *Advances in neural information processing systems*, pages 2863–2871.

[48] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in PyTorch.

[49] Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning (ICML)*, volume 2017.

[50] Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

[51] Racanière, S., Weber, T., Reichert, D., Buesing, L., Guez, A., Rezende, D. J., Badia, A. P., Vinyals, O., Heess, N., Li, Y., et al. (2017). Imagination-augmented agents for deep reinforcement learning. pages 5690–5701.

[52] Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In Xing, E. P. and Jebara, T., editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Bejing, China. PMLR.

[53] Schmidhuber, J. (1991). A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227.

[54] Shyam, P., Jaśkowski, W., and Gomez, F. (2018). Model-based active exploration. *arXiv preprint arXiv:1810.12162*.

[55] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484.

[56] Sohn, K., Lee, H., and Yan, X. (2015). Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pages 3483–3491.

[57] Stadie, B. C., Levine, S., and Abbeel, P. (2015). Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*.

[58] Sutton, R. S. (1991). Dyna, an Integrated Architecture for Learning, Planning, and Reacting. *SIGART Bull.*, 2(4):160–163.

[59] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA.

[60] Thrun, S. B. and Möller, K. (1992). Active exploration in dynamic environments. In *Advances in neural information processing systems*, pages 531–538.

[61] Van Den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. W., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. In *SSW*, page 125.

[62] van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. (2016). Pixel recurrent neural networks. In *ICML*.

[63] Wahlström, N., Schön, T. B., and Deisenroth, M. P. (2015). From Pixels to Torques: Policy Learning with Deep Dynamical Models. *arXiv:1502.02251 [cs, stat]*.

[64] Wang, T.-C., Liu, M.-Y., Zhu, J.-Y., Liu, G., Tao, A., Kautz, J., and Catanzaro, B. (2018). Video-to-video synthesis. *arXiv preprint arXiv:1808.06601*.

[65] Wang, Z., Merel, J. S., Reed, S. E., de Freitas, N., Wayne, G., and Heess, N. (2017). Robust Imitation of Diverse Behaviors. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5320–5329. Curran Associates, Inc.

[66] Watter, M., Springenberg, J., Boedecker, J., and Riedmiller, M. (2015). 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., editors, *Advances in Neural Information Processing Systems 28*, pages 2746–2754. Curran Associates, Inc.

[67] Wiering, M. and Schmidhuber, J. (1998). Efficient Model-Based Exploration. In *Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior: From Animals to Animats 6*, pages 223–228. MIT Press/Bradford Books.

[68] Yu, L., Zhang, W., Wang, J., and Yu, Y. (2016). SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. *arXiv:1609.05473 [cs]*.

[69] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, Venice. IEEE.