Generalization Across Space and Time in Reinforcement Learning

Alex Lewandowski
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada
lewandowski@ualberta.ca

Abstract

Reinforcement learning relies on generalization concepts from supervised learning. For example, generalizing to new states or environments corresponds to test-set generalization or transfer learning from supervised learning. While these concepts are applicable to reinforcement learning, they fail to consider the entire reinforcement learning problem. To address this, I characterize generalization in reinforcement learning across three axes: state-space, action-space and time. For my pre-registration proposal, I describe a method to transform a classification dataset into a contextual bandit environment and then a Markov decision process. I then propose an experiment in which I randomly corrupt labels of the classification dataset, inducing a new Markov decision process, and probe generalization across the three axes in online and offline reinforcement learning, including both model-free and model-based methods.

1 Introduction

In Reinforcement Learning (RL), an agent is tasked to maximize its sum of future reward in an unknown environment that is modelled by a Markov Decision Process (Sutton and Barto, 2018). To accomplish this, an agent learns various functions to guide its behavior, such as the value, action-value, policy, state-transition or reward-transition model. The classic tabular approach to RL learns these functions in an array or table, indexed by state and/or action. However, tabular methods do not generalize – learning at one state or action has no impact on learning at other states and actions. The goal of function approximation is to enable the agent to leverage learning from previously encountered states and actions to help the agent in yet unseen states and actions. This powerful idea has led RL, in combination with neural networks, to success in games like Atari (Mnih et al., 2013), Chess, Shogi, Go (Schrittwieser et al., 2019) and StarCraft (Vinyals et al., 2017).

The Markov Decision Process (MDP) formalism underpins task specification in RL (White, 2016; Lattimore and Szepesvári, 2020). An MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, r, \mathcal{P}, \mu, \gamma)$, where $\mathcal{A}$ denotes the action space, $\mathcal{S}$ is the state space, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function that maps a state and an action to a reward, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state transition function, μ is the initial state distribution and $\gamma \in [0, 1]$ is the discount factor. RL differs from supervised learning in three important ways. First, the feedback that the agent receives from the environment carries less information than a supervised signal because it depends on the action taken by the agent. Second, the states that an RL agent encounters are temporally correlated and also determined by the actions taken by the agent. Lastly, and most importantly for notions of generalization, the problem specification for an RL agent is generally not determined by a dataset that can be split into testing and training components. An RL agent gathers its own data according to a behavior policy π_b. Hence, it is not always clear what it means to generalize in RL.

Pre-registration workshop NeurIPS (2020), Vancouver, Canada.

In supervised learning, if a function approximator $f : \mathcal{X} \to \mathbb{R}^d$ is trained on a set $\mathcal{X}_{\text{train}}$ sampled from $\mathcal{X}$, then f generalizes well if the difference between the error on the testing set $\mathcal{X}_{\text{test}}$ and the error on the training set $\mathcal{X}_{\text{train}}$, $\mathcal{R}_{\mathcal{X}_{\text{test}}}(f) - \mathcal{R}_{\mathcal{X}_{\text{train}}}(f)$, is small. Neural networks' success can be attributed to their ability to learn complex nonlinear relationships without overfitting and impeding their ability to generalize (Zhang et al., 2016). This has led to advances in both supervised and unsupervised learning methods (Goodfellow et al., 2014; Kingma and Welling, 2013), and several theories have been put forward to partially explain the generalization capabilities of neural networks (Arora et al., 2019; Ba et al., 2020; Goldblum et al., 2019). In these settings, and in contrast to RL, practitioners have the luxury of sampling a training and testing set from one larger dataset.

In this paper, I identify three axes of generalization across state-space, action-space and time. For each axis, I summarize notions of generalization in the RL literature. I then propose an RL environment that constructs an MDP using a classification dataset, which allows the probing of an agents generalization capabilities. Based on the three aforementioned axes of generalization and the classification-based MDP environment, I investigate both model-free and model-based agents and their ability to generalize in offline and online RL through a classification-label corruption experiment (Zhang et al., 2016).

2 Generalization in Reinforcement Learning

In this section, I examine generalization in RL across three axes: state-space, action-space and time. First, I define generalization in both state and action space before discussing previous work investigating these two notions of generalization. Next, I define a new notion of generalization across time that has not been previously proposed. While state-space generalization is a well explored concept in RL, action-space generalization is less explored, with temporal generalization left unexplored. I argue that generalization in action-space and time is a unique challenge, and successful generalization across these axes is pivotal for a true synthesis of neural networks and RL.

One type of generalization that I will not discuss is generalization to new MDPs, which is closer to transfer learning from supervised learning. In particular, I do not consider generalization to changes in the MDP, such as when the state or reward transition function changes (Packer et al., 2018). The work of Cobbe et al. (2018), for example, proposes the CoinRun environment which can procedurally generates levels in a platformer game. One can then generate separate training and testing environments so that the agent can be evaluated on the hold-out set of testing environments. This notion of generalization across MDPs can be seen as generalization to unseen states if the difference between MDPs is only in the start-state distribution. In general, the generated MDPs in CoinRun can differ substantially in ways that are difficult to quantify. The classification-based environment proposed in this paper leverages a similar idea by evaluating the agent on states from a test set. The test set is not a separate environment, however, and it is only used for evaluation purposes.

2.1 State-space Generalization

State-space generalization is the most commonly explored notion of generalization in RL. Given a state s that has not seen by the agent during learning, the agent is evaluated on quantities related to that state. In model-free methods for example, an agent may be evaluated on the accuracy of its value estimate or the KL-divergence of its learned policy distribution to the true policy distribution. Model-based methods can be evaluated by their estimate of immediate reward in expectation over all actions. In online RL, this notion of generalization is not straightforward because the agent does not put weight on states outside of its current stationary distribution $d^\pi(s)$. It would be unreasonable to expect an agent to generalize to an arbitrary state. More reasonable would be to expect the agent to generalize to states near the stationary distribution $d^\pi(s)$, as changes in the policy (from learning) will change this distribution. In offline RL, the agent is expected to generalize to states reachable by its learned target policy, which is generally follows a different distribution from its training dataset.

Generalization across state-space has been previously studied with respect to an RL agent's ability to overfit to its training experience. This was investigated in Atari games (Bellemare et al., 2012), with proposals to increase stochasticity with sticky actions to minimize overfitting (Machado et al., 2017). Work by Zhang et al. (2018), however, found that agents routinely overfit by memorizing action sequences in the environment, even with sticky actions. Work by Farebrother et al. (2018)

investigates regularization methods from supervised learning, such as dropout (Gal and Ghahramani, 2015). They found that regularization can help generalization to different versions of the same Atari game. Overall, this notion of generalization is most amenable to supervised learning concepts.

2.2 Action-space Generalization

Generalization to out-of-distribution actions is a less explored notion compared to state-space generalization. To evaluate action-space generalization, the environment can be reset to a state that the agent has previously encountered. From this state, the agent is evaluated on quantities related to actions that were not taken. In model-free methods for example, the agent can be evaluated on its estimate of the action-value for an action not taken by the agent. For model-based methods, one might measure how accurate the estimate of the reward is for actions not taken. This type of generalization is arguably more important than state-space generalization because policies in RL tend to place non-zero probability on all actions for exploration purposes.

Action-space generalization has seen limited investigation. The work of Chandak et al. (2019) finds that learned action representations are able to improve generalization for large discrete action spaces. No work has investigated the interplay between learned representations from state and its effect on generalization across actions.

2.3 Temporal Generalization

I define temporal generalization as an agent's ability to predict out-of-distribution quantities that are extended in time. Model-free methods do not explicitly represent time due to the Markov assumption. However, the TD error (Sutton and Barto, 2018) encodes temporal information through differences in successive states. If a model-free agent is trained with 1-step bootstrapped returns for example, how well does the agent approximate the TD error for n-step bootstrapped returns? For model-based methods, temporal generalization is natural. If a recurrent reward-transition model is trained on sequences of length H for example, the agent's reward transition model can be evaluated on sequences of length $2H$. Temporal generalization implicitly depends on state and action generalization and, hence, is necessarily harder than them.

Generalization across time has not been explicitly explored in the literature. From the perspective of gradient interference, Bengio et al. (2020) proposed measuring interference loss during optimization and identifies a relationship between generalization and interference solutions with bootstrapping. While interesting, this notion of generalization is distinct from the work presented here which focuses on generalization.

3 Transforming Classification Problems Into MDPs

Evaluating generalization in RL is difficult in part due to the complexity of MDP environments. While there are several tabular environments that can evaluate other RL capabilities (such as exploration), striking the balance between tractability and complexity is a challenge in evaluating generalization and function approximation. Some candidate environments for assessing generalization in RL include: classic control environments (Osband et al., 2019; Brockman et al., 2016), Atari (Bellemare et al., 2012) and MuJoCo (Todorov et al., 2012). Training agents on Atari and MuJoCo is time consuming, and subject to high variance. Even for classical control experiments, such as cartpole (Barto et al., 1983) and mountain car (Moore, 1990), it is difficult to assess an agent's ability to generalize at a particular query state when the agent's stationary distribution $d^{\pi}(s)$ is unknown. Without knowing the density associated with the query state, it is possible that the query state is unreachable, or that it is sufficiently different from all previously encountered states that it is unrealistic, and unnecessary, for the agent to generalize to that state. To ease these difficulties, I propose constructing an MDP using a classification dataset.

3.1 From Classification to Contextual Bandit

A contextual bandit problem is defined by a set of states (or contexts) $\mathcal{S}$, a set of actions $\mathcal{A}$ and a reward function $r(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Any classification dataset can be transformed into a contextual bandit problem, for example the MNIST dataset is a common contextual bandit testbed (Osband et al., 2019; Chen et al., 2019). To formalize this, let the classification dataset be represented as a set of n data points $\mathcal{X} = \{x_1, \dots, x_n\}$ where each $x_i \in \mathbb{R}^d$ and labels $\mathcal{Y} = \{y_1, \dots, y_n\}$ where each $y_i \in [k]$. Then, a contextual bandit problem can be constructed by letting the set of contexts be equal to the set of data points. An agent interacts with this environment by receiving a random state $s = x_i$ and choosing an action a, *i.e.* by classifying the data point. The reward is calculated as a function of the real label of the data point y_i and the action selected a. Usually, we give the agent a binary reward of 1 if $a = y_i$ and 0 otherwise.

3.2 From Contextual Bandit to MDP

Contextual bandit environments do not have a temporal component, because each new state is randomly sampled regardless of the taken action. To construct an MDP, I must specify the structure of state transition, the reward structure, discounting, and time limits for the agent.

State transitions: The environment is initialized with a randomly chosen data point of the first label which the agent must classify. There are two types of state transitions dependent on whether the action is optimal or not. If the action is optimal, then the classification is correct, and the next state will correspond to a data point of the next label. However, if the action selected is incorrect then the next state will be from the same class. A more difficult alternative of this environment will have the agent start from the initial state if the incorrect action is chosen.

Reward structure and discounting: While the agent could still receive a reward of 1 for a correct classification and a reward of 0 otherwise, this would give an unfair advantage to the agent that estimate the reward function and selects actions myopically. To alleviate this, a reward of 1 will be given to the agent only after reaching the final label. A discount factor $\gamma = 0.99$ is also necessary so that the agent prefers to quickly reach the terminal state.

Time limits: For a dataset with N labels, I will treat the last label in the dataset as a terminating state. If an incorrect action does not reset the episode, then the expected number of steps to reach the end is N^2. If an incorrect action results in a reset, a uniformly random behavior policy will take N^N steps to successfully complete an episode. This can be prohibitively long, and so the number of labels may be limited to $M << N$. In any case, I will cut-off the episode at 200 time steps, but not marking this as termination if the last label was not reached (Pardo et al., 2017).

4 Experimental Design

To evaluate the axes of generalization outlined in Section 2, I will convert the MNIST classification dataset into an MDP. MNIST has been previously studied as a contextual bandit problem (Chen et al., 2019; Joachims et al., 2018), but I have not seen work studying it as an MDP. This will allow me to probe action-space and state-space generalization in a manner similar to a classification problem. The benefit of using this construction is that I can use hypotheses from the supervised learning literature and investigate them in the MDP setting. I will reproduce the label-corruption experiment of Zhang et al. (2016). In particular, I investigate the effect of partial label corruption where labels in the training set have a probability p of being corrupted to incorrect labels. If one of the RL methods successfully train and generalize in the partially corrupted case, I will also investigate full corruption where all labels are randomly reassigned in the training set. An experiment in similar spirit was conducted by Zhang et al. (2018), where rewards were randomized. However, this type of randomization does not systematically change state-transitions and optimal actions as it does in my environment construction.

My primary interest is the control problem, and so I will have two separate agents that learn either an action-value network or a reward-transition model. The agent using an action-value network will explore with ϵ-greedy actions and use Q-learning updates. The reward-transition agent will use a recurrent network and interact with the environment using random-shooting, which is a strong baseline (Wang et al., 2019). The recurrent model will be trained to minimize the $L2$ error between its estimates of the reward and its received rewards. My reason for not including policy gradient methods is that they do not presently lend themselves to action-space generalization (since they are defined as a mapping to a specific action or a distribution over actions) or temporal generalization (since they do not follow a Bellman equation).

I will investigate generalization in both online and offline RL due to possible discrepancies between the two training regimes (Agarwal et al., 2019). Similar to work by Fu et al. (2020), offline training will consist of three different regimes corresponding to the initialized buffer, the replay buffer half way through training and the replay buffer after training.

4.1 Experiment Details

Generalization metrics: Let $f(s,a)$ denote the agent's $Q(s,a)$ or $r(s,a)$ for model-free or model-based respectively, similarly $f^*(s,a)$ for the true reward / optimal value. For a policy π, denote the stationary distribution as $d^\pi(s)$. Then my MDP construction allows us to partition the state-space into i.i.d. samples from $d^\pi(s)$, S_{train} and S_{test}. Denote the set of states that share the label of state s as $S_{train}(s)$. The states of the replay buffer D are a subset of the training partition, $D_s \subset S_{train}$. Given s, a from the replay buffer, I sample a state with the same label from the test set $s' \sim S_{test}$. The state-space generalization metric compares the optimal quantity with the predicted quantity at the test state s' after taking the training action a, $M_{state}(f, D) = \mathbb{E}_{s,a\sim D,\, s'\sim S_{test}(s)}\left[\|f^*(s',a) - f(s',a)\|^2\right]$. For action-space generalization, I consider the difference between the optimum quantity and the prediction at the test state s' for the worst-case action, $M_{action}(f, D) = \mathbb{E}_{s\sim D, s'\sim S_{test}(s)}\left[\max_{a'} \|f^*(s',a') - f(s',a')\|^2\right]$.

The generalization metrics for state-space and action-space evaluate 1-step rewards or returns. I now extend this to generalization across time. Let $\{s_t, a_t\}_{t=1}^H$ be a state-action sequence of length H. The model-based agent is trained to predict the next H rewards and the model-free agent uses H-step returns. Temporal generalization evaluates both agents on sequences of length $2H$ where the first H state-actions s_t, a_t are a sequence from the replay-buffer, and s'_t is sampled from S_{test} with the same label as s_t. The remainder of the sequence is simulated in the test environment starting at s'_H and with randomly selected actions $\{a_{t+H}\}_{t=1}^H$. The metric for evaluating the temporal generalization of an agent is thus, $M_{time}(f, D) = \mathbb{E}\left[\sum_{t=1}^{2H} \|f^*(s_t,a_t) - f(s_t,a_t)\|^2\right]$. For model-free methods, this metric does not explicitly represent time and instead represents further out-of-distribution state and action generalization. If this metric is insufficient, then one can also compare TD errors on the relevant horizons: $\|f(s_1,a_1) - \gamma^{t-1}\max_{a'} f(s_{2H},a') - \sum_{t=2}^{2H}\gamma^{t-2} r(s_t,a_t\|^2$. This is less justified, because it is not proven that Q-learning minimizes this objective. Furthermore, there is no optimal target (f^*) for comparison. Nevertheless, this quantity does capture a notion of temporal generalization in model-free learning.

Online training protocol: The online agent initializes a replay buffer with 10 trajectories from a random policy before proceeding to interleave environment interaction with parameter updates. Training is concluded when the agent solves the environment 10 times in a row.

Offline training protocol: The offline agent uses the replay buffer of the online agent at one of three stages: initialization, half-way through training and after training has concluded. The agent uses this fixed dataset to optimize the reward model or action-value until training converges.

Hyperparameters: For all models, I use ADAM and learning rates will be swept over $\alpha \in \{0.5^n\}_{n=1}^{10}$. Labels will be corrupted according to $p \in \{0.0, 0.5, 1.0\}$. For model-free methods, $\epsilon = 0.1$. The reward model predicts rewards $H = 5$ steps into the future and the model-free agent uses 5-step returns.

5 Hypotheses and Possible Conclusions

In general, my hypothesis is that generalization is more difficult as the problem moves from supervised learning to the MDP. I hypothesize that model-free and model-based methods are able to generalize equally well in state-space. However, I suspect that model-based methods will generalize better than model-free methods in action-space and time.

Model-free generalization: I hypothesize that model-free methods will successfully generalize to unseen states corresponding to seen labels, because they correspond to the same underlying state. However, I suspect that they will be unable to generalize to unseen labels. For action-space generalization, I expect that model-free methods will be able to generalize to out-of-distribution actions, but not as well as the model-based agent. This is because the state is processed as an input to the network, whereas discrete actions are indexes on the output of the network. This would diminish a network's ability to generalize to other actions. Lastly, I hypothesize that generalization across time to worsen greatly as the number of bootstrapping steps increases. When labels are corrupted, I hypothesize that model-free methods will not be able to fit the experience from the label-corrupted MDP.

Model-based generalization: Estimating transition models in model-based RL is essentially a supervised learning problem. Hence, I expect that transition models will generalize well across the three different axes. Specifically, given an unseen state of a seen label, both state and reward transition models will successfully generalize. However, I predict that temporal generalization will suffer for longer horizons but not to the extent of model-free methods. I would not expect transition models to generalize to unseen labels, as I would not expect a supervised learning model to do 0-shot generalization without explicit (meta-) training. When labels are corrupted, I hypothesize that transition models will be able to fit the corrupted experience.

References

Agarwal, R., Schuurmans, D., and Norouzi, M. (2019). An optimistic perspective on offline reinforcement learning. *arXiv:1907.04543*.

Arora, S., Du, S. S., Hu, W., Li, Z., and Wang, R. (2019). Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. *arXiv:1901.08584*.

Ba, J., Erdogdu, M., Suzuki, T., Wu, D., and Zhang, T. (2020). Generalization of two-layer neural networks: An asymptotic viewpoint. In *International Conference on Learning Representations*.

Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2012). The arcade learning environment: An evaluation platform for general agents. *arXiv:1207.4708*.

Bengio, E., Pineau, J., and Precup, D. (2020). Interference and generalization in temporal difference learning. *arXiv:2003.06350*.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym.

Chandak, Y., Theocharous, G., Kostas, J., Jordan, S., and Thomas, P. S. (2019). Learning action representations for reinforcement learning. *arXiv:1902.00183*.

Chen, M., Gummadi, R., Harris, C., and Schuurmans, D. (2019). Surrogate objectives for batch policy optimization in one-step decision making. In Wallach, H., Larochelle, H., Beygelzimer, A., dÁlché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8825–8835. Curran Associates, Inc.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. (2018). Quantifying generalization in reinforcement learning. *arXiv:1812.02341*.

Farebrother, J., Machado, M. C., and Bowling, M. (2018). Generalization and regularization in dqn. *arXiv:1810.00123*.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. (2020). D4rl: Datasets for deep data-driven reinforcement learning. *arXiv:2004.07219*.

Gal, Y. and Ghahramani, Z. (2015). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *arXiv:1506.02142*.

Goldblum, M., Geiping, J., Schwarzschild, A., Moeller, M., and Goldstein, T. (2019). Truth or backpropaganda? an empirical investigation of deep learning theory. *arXiv:1910.00359*.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks. *arXiv:1406.2661*.

Joachims, T., Swaminathan, A., and de Rijke, M. (2018). Deep learning with logged bandit feedback. In *International Conference on Learning Representations*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv:1312.6114*.

Lattimore, T. and Szepesvári, C. (2020). *Bandit Algorithms*. Cambridge University Press.

Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M., and Bowling, M. (2017). Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *arXiv:1709.06009*.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv:1312.5602*.

Moore, A. W. (1990). Efficient memory-based learning for robot control. Technical report, University of Cambridge.

Osband, I., Doron, Y., Hessel, M., Aslanides, J., Sezener, E., Saraiva, A., McKinney, K., Lattimore, T., Szepezvari, C., Singh, S., Roy, B. V., Sutton, R., Silver, D., and Hasselt, H. V. (2019). Behaviour suite for reinforcement learning. *arXiv:1908.03568*.

Packer, C., Gao, K., Kos, J., Krähenbühl, P., Koltun, V., and Song, D. (2018). Assessing generalization in deep reinforcement learning. *arXiv:1810.12282*.

Pardo, F., Tavakoli, A., Levdik, V., and Kormushev, P. (2017). Time limits in reinforcement learning. *arXiv:1712.00*.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T., and Silver, D. (2019). Mastering atari, go, chess and shogi by planning with a learned model. *arXiv:1911.08265*.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press.

Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE.

Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J., Schrittwieser, J., Quan, J., Gaffney, S., Petersen, S., Simonyan, K., Schaul, T., van Hasselt, H., Silver, D., Lillicrap, T. P., Calderone, K., Keet, P., Brunasso, A., Lawrence, D., Ekermo, A., Repp, J., and Tsing, R. (2017). Starcraft II: A new challenge for reinforcement learning. *arXiv:1708.04782*, abs/1708.04782.

Wang, T., Bao, X., Clavera, I., Hoang, J., Wen, Y., Langlois, E., Zhang, S., Zhang, G., Abbeel, P., and Ba, J. (2019). Benchmarking model-based reinforcement learning. *arXiv:1907.02057*.

White, M. (2016). Unifying task specification in reinforcement learning. *arXiv:1609.01995*.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *arXiv:1611.03530*.

Zhang, C., Vinyals, O., Munos, R., and Bengio, S. (2018). A study on overfitting in deep reinforcement learning. *arXiv:1804.06893*.