GENERALIZING FROM A FEW ENVIRONMENTS IN SAFETY-CRITICAL REINFORCEMENT LEARNING

Zachary Kenton[1] **Angelos Filos**[1] **Yarin Gal**[1] **Owain Evans**[2]
[1]University of Oxford [2]Future of Humanity Institute

ABSTRACT

Before deploying autonomous agents in the real world, we need to be confident they will perform safely in novel situations. Ideally, we would expose agents to a very wide range of situations during training (e.g. many simulated environments), allowing them to learn about every possible danger. But this is often impractical: simulations may fail to capture the full range of situations and may differ subtly from reality. This paper investigates generalizing from a limited number of training environments in deep reinforcement learning in which safety is critical. Our experiments test whether agents can perform safely in novel environments, given varying numbers of environments at train time. Using a gridworld setting, we find that standard deep RL agents do not reliably avoid catastrophes on unseen environments – even after performing near optimally on 1000 training environments. However, we show that catastrophes can be significantly reduced (but not eliminated) with simple modifications, including Q-network ensembling to represent uncertainty and the use of a classifier trained to recognize dangerous actions.

1 INTRODUCTION

Problem Setting. Recent progress in deep reinforcement learning (RL) has achieved impressive results in a range of applications from playing games (Mnih et al., 2015; Silver et al., 2016), to dialogue systems (Li et al., 2017) and robotics (Levine et al., 2016; Andrychowicz et al., 2018). However, generalizing to unseen environments remains difficult for deep RL algorithms, which can fail catastrophically when encountering new environments (Leike et al., 2017). We consider the setting where an RL agent trains on a limited number of environments and must generalize to unseen environments. The agent will not perform perfectly on the unseen environments. But can it avoid dangers that were already encountered during training?

In safety-critical domains, catastrophic outcomes are unacceptable (Garcıa & Fernández, 2015). RL agents would ideally be able to avoid persistent dangers, without requiring hand-crafting of a safe policy.

In this work, we assume that we have access to a simulator, which captures the basic semantics of the world (i.e. dangers, goals and dynamics). In the simulator the agent can experience dangers and learn from them (Paul et al., 2018). We evaluate agents on how well they can transfer knowledge: can they generalize to unseen environments with the same basic semantics? At deployment, the agent has a single episode to solve an unseen environment and any dangerous behaviour is unacceptable.

Related Work. Motivated by the standard regularization methods for tackling overfitting in deep neural networks, Farebrother et al. (2018) and Cobbe et al. (2018) experiment with L2-regularisation, dropout (Srivastava et al., 2014) and batch normalization (Ioffe & Szegedy, 2015) with Deep Q-Networks (Mnih et al., 2015), showing improved generalization performance.

Zhang et al. (2018) investigate the ability of A3C (Mnih et al., 2016) to generalize rather than memorize in a set of gridworlds similar to our environments. They show that perfect generalization is possible when a sufficient amount of environments is provided (10000 environments), but they do not focus on the regime of a limited number of training environments, nor evaluate performance in terms of safety. At the other extreme, Leike et al. (2017) introduce a 'Distribution Shift' gridworld setup, where they train on a single environment and deploy on another.

In a different direction, Saunders et al. (2018) approached danger avoidance by using supervised learning to train a blocker (i.e. a classifier) using a human-in-the-loop to maintain safety during training, which restricts its scalability. A collision prediction model was also considered in the model-based setting in Kahn et al. (2017). In Lipton et al. (2016), catastrophes are avoided by training an intrinsic fear model to predict whether a catastrophe will occur, and using this to perform reward shaping.

From a modeling perspective, an ensemble of models often performs better than a single model (Dietterich, 2000). They can also be used for predictive uncertainty estimation of deep neural networks (Lakshminarayanan et al., 2017). We make use of this in our safety-critical investigation.

Finally, our approach can also be related to meta-learning (Schmidhuber, 1987; Thrun & Pratt, 2012; Hochreiter et al., 2001; Bengio et al., 1992), which is concerned with learning strategies which are fast to adapt using prior experience. In the RL context, approaches include gradient-based (Finn et al., 2017) and recurrent style (Wang et al., 2016; Duan et al., 2016) models using multiple environments to train from. Our setting corresponds to the zero-shot meta-RL setting, in which we train on multiple training environments but do not adapt based on test environment reward signals.

Contributions. We introduce a new class of gridworlds for evaluating whether agents can generalize catastrophe-avoidance from training environments to unseen test environments. We find that standard DQN fails to avoid catastrophes at test time, even with 1000 training environments. We compare standard DQN to modified versions that incorporate dropout, Q-network ensembling, and a classifier to recognize dangerous actions. These modifications reduce catastrophes significantly, including in the regime of very few training environments.

2 BACKGROUND

Task Setup. We consider an agent interacting with an environment in the standard RL framework (Sutton & Barto, 2018). At each step, the agent selects an action based on its current state, and the environment provides a reward and the next state. Our task setup is the same as in (Zhang et al., 2018): there is a train/test split for *environments* that is analogous to the train/test split for *data points* in supervised learning. In our experiments all environments will have the same reward and transition function, and differ only in how entities are arranged in the gridworld (the initial state). Hence we can equivalently describe our setup in terms of a distribution on initial states for a single MDP.

Formally, we denote our task by $(\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, R), \mathcal{P}_0)$, where $\mathcal{M}$ is a Markov Decision Process (MDP), with state space $\mathcal{S}$, action space $\mathcal{A}$, transition probability $\mathcal{P}$ and immediate reward function r. Additionally, $\mathcal{P}_0$ is a probability distribution on the initial state $\mathcal{S}_0 \subset \mathcal{S}$. We use the undiscounted episodic setting, where each episode randomly samples an initial state from $\mathcal{P}_0$ and ends in a finite number of timesteps, T. There are disjoint training and test sets which have i.i.d. samples $\hat{\mathcal{S}}_0$ from $\mathcal{P}_0$. During training the agent encounters initial states only from the training set and makes learning updates based on the observed rewards. Test performance is calculated on the test set, and no learning takes place at test time.

An agent follows policy $\pi : \mathcal{S} \times \mathcal{A}$ and the corresponding action-value is defined by $Q^\pi(s, a) = \mathbb{E}[\sum_{t=0}^{T} r_t | s_t = s, a_t = a]$, for an action a, and state s, where the expectation is taken over trajectories following policy π. The optimal action-value function is $Q^*(s, a) = \max_\pi Q^\pi(s, a)$, and obeys the Bellman optimality equation $Q^*(s, a) = \mathbb{E}_{s'}[r + \max'_a Q^*(s', a')]$. Q-learning (Watkins & Dayan, 1992) is a standard algorithm for finding the optimal Q^* value for a finite MDP, when both the state and the action space are finite sets. In order to generalize across MDPs (or to unseen states in a single large MDP), we employ Q-learning with function approximation (Sutton & Barto, 2018).

Deep Q-Networks (DQN). Deep Q-networks (Mnih et al., 2015) use a deep neural network, $Q(s, a; \theta)$, with parameter vector θ, to estimate the optimal value function. DQN is optimized by minimizing $L_i(\theta_i) = \mathbb{E}_{s,a,r,s'}[(y_i - Q(s, a; \theta_i))^2]$, at each iteration i, where $y_i = r + \max_{a'} Q(s', a'; \theta^-)$. The θ^- are parameters of a target network that is kept frozen for a number of iterations whilst updating the online network parameters θ. The optimization is performed off-policy, randomly sampling from an experience replay buffer. During training, actions are chosen using the ϵ-greedy exploration

strategy, selecting a random action with probability ϵ and otherwise taking the greedy action (which has maximum Q-value). At test time, the agent acts greedily.

Model Averaging. Ensembles of models (i.e. *model averaging*) are usually used for estimating model (i.e. *epistemic*) uncertainty. In particular, instead of a single model f, a set of models $f_1, f_2, \ldots, f_N$ is fitted. Then either the average, $f_{\text{ens}} = \frac{1}{N}\sum_{n=1}^{N} f_n$ or, in classification tasks, the mode (i.e. *majority vote*) $f_{\text{maj}} = \texttt{mode}(f_1, f_2, \ldots, f_N)$ is used for prediction. When neural networks are used as models, the diversification between the models is obtained by initializing them differently and by following independent training.

Catastrophe Classifier. Another approach to avoiding dangers is to learn a classifier for whether an action-state pair will be catastrophic and use this to block certain actions — see (Saunders et al., 2018) for an example trained with a human-in-the-loop. During training we store all state-action pairs, together with a binary label of whether a catastrophe occurred. Then after training the DQN agent, we separately train the classifier to predict the probability that a state-action pair will result in a catastrophe. Training is done in a supervised manner by minimizing the binary cross entropy loss. The classifier is used as a 'blocker' at deployment time. At test time we run our selected action through the classifier and block the action if the classifier predicts it is catastrophic with confidence greater than some threshold. We then move on to the next highest value action and run that through the classifier. The process repeats until an acceptable action is found, otherwise the episode is terminated. Note that the blocker will only block dangerous actions that occur just before the danger is about to be experienced, but won't help for those actions which irreversibly cause a catastrophe to occur many steps later (Saunders et al., 2018).

3 Experiments

Experimental Setup. Our environment setup is a distribution of gridworld environments, each of which is size 5×5, and contains an agent (blue), a single lava cell (red) and a single goal cell (green). The agent receives sparse rewards of $+1$ for reaching the goal and -1 for reaching the lava. The episode terminates whenever the goal or lava is reached, or when fifty timesteps have elapsed (giving zero reward), whichever occurs first. We consider two environment settings, which we call `Full` and `Reveal`. In `Full`, the agent sees the full map — an example trajectory is shown in Fig. 1. In `Reveal`, Fig. 2, the agent starts off seeing only part of the map, and reveals the map as it goes around, with a 3×3 view. `Reveal` is a more challenging setting because it requires the agent to move around to uncover the position of the goal. The agent receives the observation an array of RGB pixel values flattened across the channel dimension.

We treat moving onto the lava as a catastrophe. Our evaluation metrics are the percentage of environments that are solved (the agent reaches the goal before the timeout), and the percentage of environments that end in catastrophe (the agent reaches the lava). On test environments we consider timeouts to be an acceptable failure, whereas a catastrophe is unacceptable.

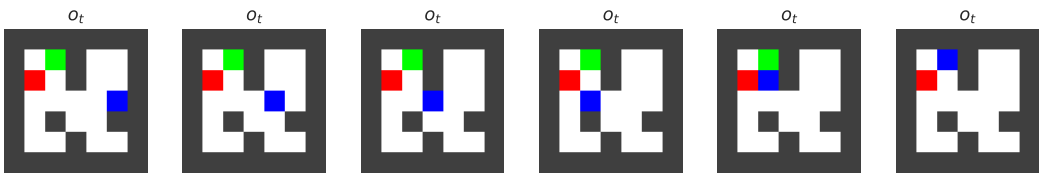


Figure 1: Example trajectory from a `Full` environment. Agent: blue. Goal: green. Lava: red. Walls: grey.

Algorithm Settings A summary of the methods used can be found in Tab. 1. All 3-layer multi-layer perceptron DQN models were trained for 1M training episodes using: hidden layer sizes [256,256,512], batch size 32, RMSProp (Tieleman & Hinton, 2012) with learning rate 1e–4, a replay buffer with 10K capacity and the target network was updated every 1K episodes. An ϵ-greedy policy was used with decay rate 0.999 and end value 0.05. The blocker is also a 3-layer multi-layer

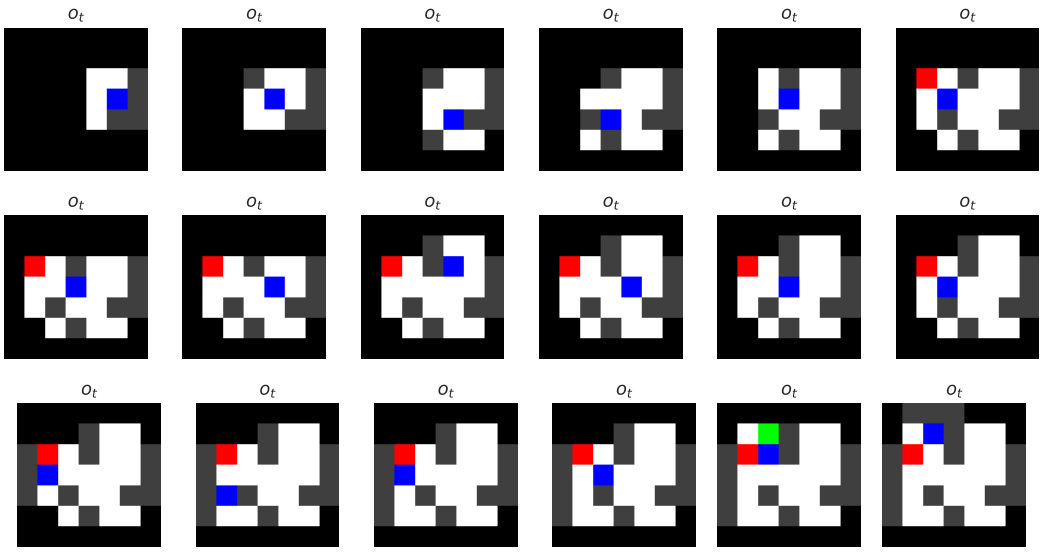


Figure 2: Example trajectory from a `Reveal` environment. Agent: blue. Goal: green. Lava: red. Walls: grey. Mask: black.

Method	**Description**
DQN	Same as (Mnih et al., 2015)
Drop-DQN	Regularized linear layers with dropout probability $p = 0.2$
Block-DQN	Catastrophe classifier used along with DQN
Ens-DQN	Ensemble of 9 independently trained and differently intialized DQNs
Maj-DQN	Majority vote of 9 independently trained and differently intialized DQNs
Block&Ens-DQN	Combination of Block-DQN and Ens-DQN

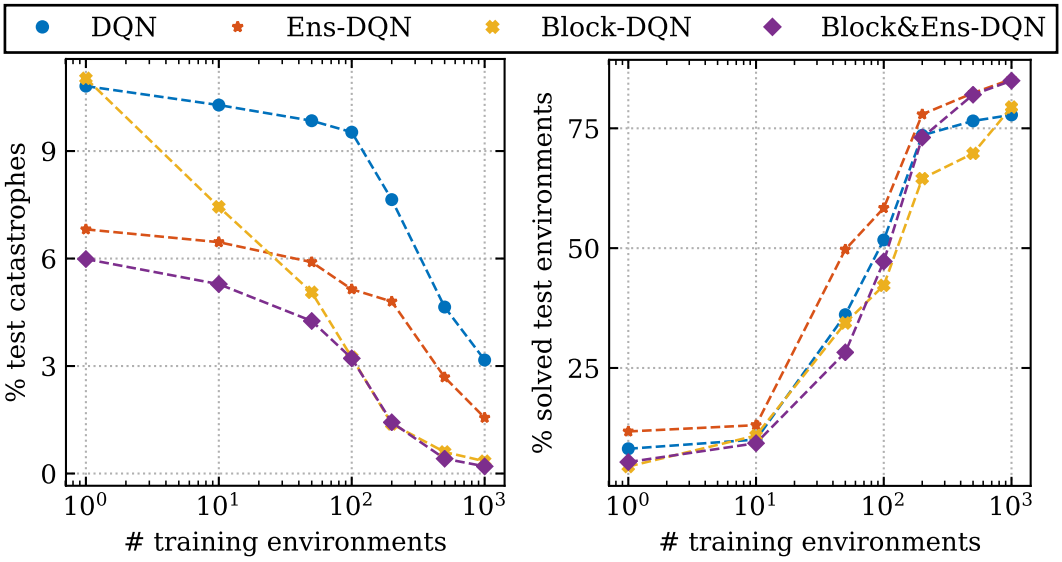
Table 1: Description of methods used in our experiments.

perceptron with hidden layer sizes [128,256,256] trained for 10k iterations using: batch size 64, Adam optimizer (Kingma & Ba, 2014) with learning rate 5e–3.

Results and Discussion. To make figures easier to read, this section includes only four methods: DQN, Ens-DQN, Block-DQN and Block&Ens-DQN. In Fig. 3 we present results on the `Reveal` gridworld. We plot the percentage of environments that ended in catastrophe in Fig. 3a, and the percentage of solved environments in Fig. 3b, as a function of the number of training environments available during training. We trained all models to convergence on the training environments. See Fig. 5 and Fig. 6 for results of all methods on `Full` and `Reveal` settings and also for the evaluations on the training environments.

Fig. 3b shows that our agents never achieve perfect performance on the test environments. Moreover, when an agent fails to reach the goal, it does not always fail gracefully (e.g. by simply timing out) but instead often ends in catastrophe (visiting the lava).

Most of the methods we investigated outperformed the DQN baseline in terms of percentage of test catastrophes. Each method offers a different trade-off between test performance on catastrophes and solved environments. For example, Block-DQN offers better catastrophe performance than DQN, but its performance on solving environments is worse given more than 100 training environments. This is possibly because the blocker is over-cautious, with too high a false-positive rate for catastrophes, which prematurely stops some environments from being solved. Note that in a real-world setting, avoiding catastrophes (Fig. 3a) will be much more important than doing well on most environments (Fig. 3b).



(a) Percentage of catastrophic outcomes in unseen environments (lower is better), as a function of number of training environments.

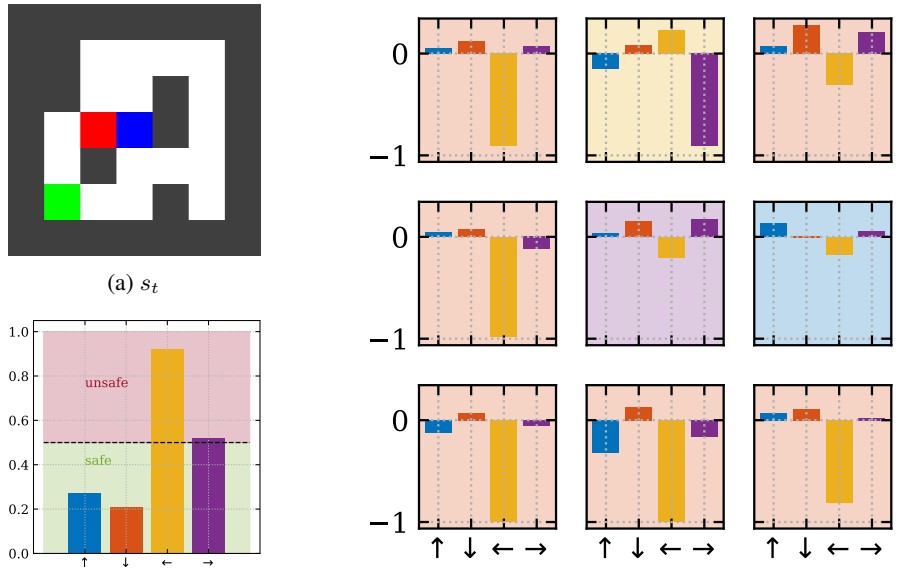
(b) Percentage of solved unseen environments (higher is better), as a function of training environments.

Figure 3: Results on the `Reveal` setting, evaluated on unseen *test* environments for a range of methods. Nine random seeds are used for each algorithm and mean performances is shown here. Figure **(a)** shows that modified algorithms outperform the baseline DQN in terms of danger avoidance. The effect on return performance is observed in **(b)**. The complete version is provided in Figure 6 of the appendix, and includes both train and test performances.

In Fig. 4 we showcase a real example from our experiments highlighting the role of the ensemble and the blocker in avoiding the catastrophe.

4 Conclusion

In this paper we investigated how safety performance generalizes when deployed on unseen test environments drawn from the same distribution of environments seen during training, where no further learning is allowed. We focused on the realistic case in which there are a limited number of training environments. We found DQN can fail dangerously on the test environments even when performing perfectly during training. We investigated some simple ways to improve safety generalization performance. Future work will build upon this paper to address adaptation to unseen test environments, allowing learning at test time, using a data-driven safe prior learnt during training.



(a) s_t

(b) $p_{\text{unsafe}}(a_t|s_t)$

(c) $Q^{(i)}(s_t, a_t)$, for $i = 1, 2, \ldots, 9$

Figure 4: Example transition by the Block&Ens-DQN in one unseen environment, in the `Full` setting. **(a)** the environment state, s_t; **(b)** the output of the trained catastrophe classifier (i.e. *blocker*) $p_{\text{unsafe}}(\cdot|s_t)$ conditioned on the environment state, where a threshold 0.5 is selected; **(c)** the nine estimates of the state-action value function $Q^{(i)}(s_t, a_t)$, for $i = 1, 2, \cdots, 9$, from the differently initialized and independently trained DQNs. The background colour highlights action with maximum value. The agent should not make the catastrophic action of going *left*, something that both the blocker and the ensemble (i.e. *model average*) of the DQNs will avoid. However, if the middle top agent in **(c)** was acting alone it would choose to go left, which would lead to a catastrophic outcome.

REFERENCES

Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018.

Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. On the optimization of a synaptic learning rule. In *Preprints Conf. Optimality in Artificial and Biological Neural Networks*, pp. 6–8. Univ. of Texas, 1992.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. *arXiv preprint arXiv:1812.02341*, 2018.

Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pp. 1–15. Springer, 2000.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Jesse Farebrother, Marlos C Machado, and Michael Bowling. Generalization and regularization in dqn. *arXiv preprint arXiv:1810.00123*, 2018.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135. JMLR. org, 2017.

Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

Sepp Hochreiter, A Steven Younger, and Peter R Conwell. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*, pp. 87–94. Springer, 2001.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Gregory Kahn, Adam Villaflor, Vitchyr Pong, Pieter Abbeel, and Sergey Levine. Uncertainty-aware reinforcement learning for collision avoidance. *arXiv preprint arXiv:1702.01182*, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pp. 6402–6413, 2017.

Jan Leike, Miljan Martic, Victoria Krakovna, Pedro A Ortega, Tom Everitt, Andrew Lefrancq, Laurent Orseau, and Shane Legg. Ai safety gridworlds. *arXiv preprint arXiv:1711.09883*, 2017.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008*, 2017.

Zachary C Lipton, Jianfeng Gao, Lihong Li, Jianshu Chen, and Li Deng. Combating reinforcement learning's sisyphean curse with intrinsic fear.(nov. 2016). *arXiv preprint cs.LG/1611.01211*, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

Supratik Paul, Michael A Osborne, and Shimon Whiteson. Fingerprint policy optimisation for robust reinforcement learning. *arXiv preprint arXiv:1805.10662*, 2018.

William Saunders, Girish Sastry, Andreas Stuhlmueller, and Owain Evans. Trial without error: Towards safe reinforcement learning via human intervention. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2067–2069. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

Jürgen Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

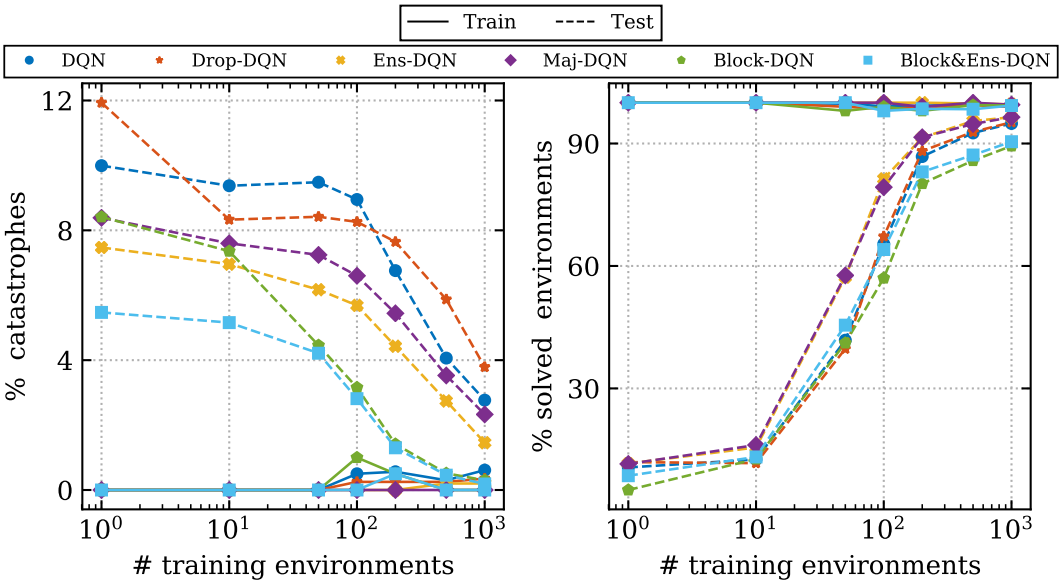
Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018.

A Appendix

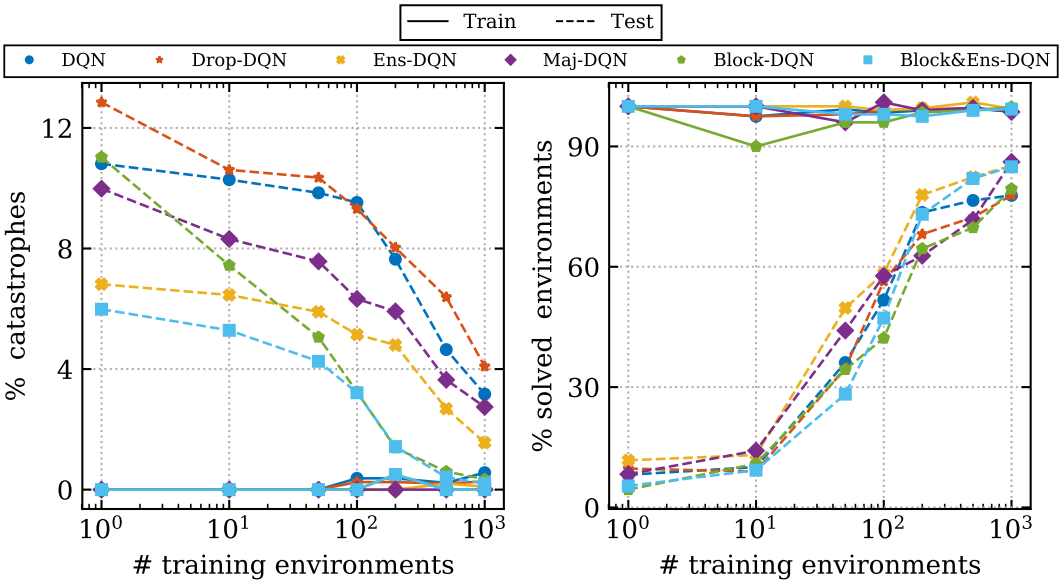
Further Results. Shown in Fig. 5 are the results for all the methods on the `Full` setting. See Fig. 6 for results on the `Reveal` setting. Shown are also the performance on the training environments (solid lines). We see similar results to the main paper, and note that as expected the `Full` setting has better generalization performance for % solved than `Reveal`, but the catastrophe performance is similar in each.



(a) Percentage of catastrophic outcomes (lower is better), as a function of number of training environments.

(b) Percentage of solved environments (higher is better), as a function of number of training environments.

Figure 5: Complete quantitative experimental results on the `Full` setting, trained to convergence. Nine seeds are used for training the agents and the mean performances are visualized.



(a) Percentage of catastrophic outcomes (lower is better), as a function of number of training environments.

(b) Percentage of solved environments (higher is better), as a function of number of training environments.

Figure 6: Complete quantitative experimental results on the `Reveal` setting, trained to convergence. Nine seeds are used for training the agents and the mean performances are visualized.