Adversarial Cheap Talk

Anonymous Authors[1]

Abstract

Adversarial attacks in reinforcement learning (RL) often assume highly-privileged access to the learning agent's parameters, environment or data. Instead, this paper proposes a novel adversarial setting called a *Cheap Talk MDP* in which an Adversary has a *minimal range of influence* over the Victim. Parameterised as a deterministic policy that only conditions on the current state, an Adversary can merely *append* information to a Victim's observation. To motivate the minimum-viability, we prove that in this setting the Adversary cannot occlude the ground truth, influence the underlying dynamics of the environment, introduce non-stationarity, add stochasticity, see the Victim's actions, or access their parameters. Additionally, we present a novel meta-learning algorithm to train the Adversary, called adversarial cheap talk (ACT). Using ACT, we demonstrate that the resulting Adversary still manages to influence the Victim's *training and test* performance despite these restrictive assumptions. Affecting train-time performance reveals a new attack vector and provides insight into the success and failure modes of existing RL algorithms. More specifically, we show that an ACT Adversary is capable of *harming* performance by interfering with the learner's function approximation and *helping* the Victim's performance by appending useful features. Finally, we demonstrate that an ACT Adversary can append information during train-time to directly and arbitrarily control the Victim at test-time in a zero-shot manner.

1. Introduction

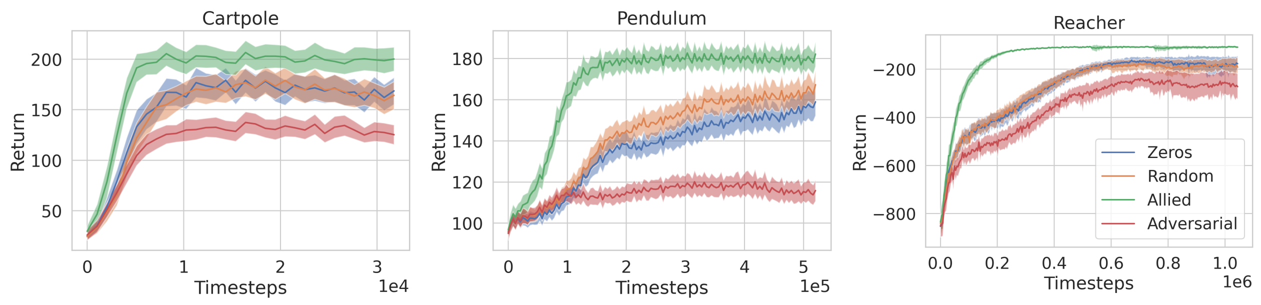
Learning agents are often trained in settings where adversaries may have some control over part of the agent's observations. However, the adversary cannot usually influence the dynamics of the underlying environment or the reward signal (at least not without cost). For example, it is often possible to append arbitrary tags to content which is later used to train recommender systems. Similarly, an adversary could rent space on interactive bulletin boards near busy traffic intersections to influence data sets which are used for training self-driving cars. Another instance occurs in financial markets, where an adversary can change the state of the order-book by submitting orders far *out of the money*. While all of these are examples of *useless* features from an information point of view, under the current paradigm of end-to-end deep learning it is common practice to include a superset of useful features as part of the input and to let the model learn which features actually matter. This paper demonstrates that an actor can heavily influence the behaviour and performance of learning agents by controlling information *only* in these "useless" features.

Most past work in adversarial attacks assumes that the adversary can influence the environment dynamics (Huang et al., 2017; Gleave et al., 2020). For example, perturbing images and observations could be used to obscure or alter relevant information, such as the ball's location in a Pong game (Kos and Song, 2017). Furthermore, many attacks require access to the trained agent's weights and parameters to generate the adversarial inputs (Wang et al., 2021). Finally, most of these attacks only cause the victim's policy to fail arbitrarily instead of giving the adversary full control over the victim's policy at test time (Gu et al., 2017; Kiourti et al., 2020; Salem et al., 2020; Ashcraft and Karra, 2021).

In contrast, our work, which is inspired by recent advancements in the field of opponent shaping (Lu et al., 2022), in Section 3 proposes a novel, minimum-viable setting to shape a learning agent, called "Cheap Talk MDP". In this setting, the Adversary can only *append* information to the observation of a Victim as a deterministic function of the current state. The Adversary does not have access to the Victim's parameters, actions, or even samples from the Victim's policy. In Section 4, we prove that the Adversary cannot change the dynamics of the underlying environment nor alter the reward functions. Nor can it inject stochasticity into the environment (deterministic) or introduce non-stationarity (function of the current state only). Furthermore, we prove that Adversaries *cannot* influence *tabular* Victims in this

[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.



(a) (b) (c)

Figure 1: Visualizations of the training curves of the Victim across different number of dimensions of the messages for (a) Cartpole, (b) Pendulum, and (c) Reacher. Error bars denote the standard error across 10 seeds of Victims trained against a single trained Adversary.

setting in Proposition 1; Adversaries can therefore only interfere with a Victim through their function approximator. In this sense, our setting represents a *bare minimum* range of influence, as further justified in Appendix B.3.

In Section 4, we also introduce a new meta-learning algorithm to train the Adversary, called adversarial cheap talk (ACT). With an extensive set of experiments in Section 5, we demonstrate that an ACT Adversary can influence a Victim to achieve a number of outcomes:

1. We show that the ACT Adversary can prevent the Victim from solving a task, resulting in low rewards *during training*. We provide empirical evidence that the Adversary sends messages which induce *catastrophic interference* in the Victim's neural network.

2. Conversely, an ACT Adversary can learn to send useful messages that *improve* the Victim's training process, resulting in higher rewards *during training*.

3. Finally, we introduce a training scheme that allows the ACT Adversary to arbitrarily control the Victim *at test-time*, in a *zero-shot* manner.

Related Work For an in-depth discussion on related work, we point the reader to Appendix A. Next, we continue by introducing the basic concepts and algorithms to understand our contributions.

2. Background

2.1. Reinforcement Learning

A Markov decision process (MDP) consists of a tuple $\mathcal{D} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ represents the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ denotes the state transition probability function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function and $\gamma \in [0, 1)$ denotes the discount factor. At every timestep t, an agent samples an action from its stochastic policy $a_t \sim \pi_\theta(\cdot \mid s_t)$, where $a_t \in \mathcal{A}$, $s_t \in \mathcal{S}$ and θ denotes the policy parameterization. The agent then receives a reward based on the action taken in the current state: $r_t = \mathcal{R}(s_t, a_t)$. Finally, a new state is sampled according to the transition function $s_{t+1} \sim \mathcal{P}(\cdot \mid s_t, a_t)$, resulting in a trajectory $\tau_\theta := ((s_0, a_0, r_0), (s_1, a_1, r_1), \ldots)$. The agent's goal is to maximize its expected discounted return under policy π_θ:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]. \tag{1}$$

3. Problem Setting

In this work, we consider two agents that interact in a setting we call a *Cheap Talk MDP* $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mathcal{M}, f, \mathcal{J} \rangle$. Here $\mathcal{M}$ denotes the space of messages. We refer to the agent observing the message as the *Victim* with transition and reward functions $\mathcal{P}, \mathcal{R}$ independent from $\mathcal{M}$. The agent appending the message is called the *Adversary*, endowed with a deterministic policy (function) $f : \mathcal{S} \to \mathcal{M}$ to append messages and an objective function $\mathcal{J}$ to optimise (details below).

The Victim is a "standard" reinforcement learning agent, selecting actions according to its policy $a_t \sim \pi_\theta(\cdot \mid s, f(s))$, where $a \in \mathcal{A}, s \in \mathcal{S}$. The Victim optimises its policy π_θ with respect to parameters θ, to maximise its expected return J defined in Equation 1.

By contrast, the Adversary may only act by appending a message $f_\phi(s)$ to s at every step, where $f_\phi : \mathcal{S} \to \mathcal{M}$ is a deterministic policy (function) of the current state and ϕ are the Adversary's parameters. These parameters may only be updated *between* full training / testing episodes of the Victim; the function remains static during episodes to avoid introducing non-stationarity. The Adversary's objective function $\mathcal{J}$ may be picked arbitrarily, and need not even be differentiable if it is optimised using ES.

In the train-time setting we focus on both the allied setting, where Adversary and Victim objectives are equal, $\mathcal{J} = J$, and the adversarial setting where objectives are zero-sum, $\mathcal{J} = -J$. In the test-time setting we use an entirely different objective, such as reaching for an arbitrary circle in Reacher (see Figure 3c). This incentivises the Adversary to manipulate the Victim into maximising $\mathcal{J}$, even if at the cost of the Victim's original objective J.

4. Method

4.1. Meta-Training Procedure

Our method treats the problem setting as a meta-learning problem. The Adversary's parameters ϕ are only updated after *a full training (and testing) run* of the Victim's parameters θ. Note that ϕ is *static* during the whole training run (inner loop) of θ and only gets updated once the inner loop completes. In the outer loop, we optimise the Adversary's objective $\mathcal{J}$ with respect to ϕ using ES as a black-box optimization technique.

4.2. Train-Time Manipulation

When influencing the agent's performance during train-time, we consider the allied and the adversarial settings. Pseudocode is provided in Algorithm 1 (see Appendix D, where E is the number of Victim training episodes and N is the ES population size. Letting $c = 1$ for allied and $c = -1$ for adversarial, the Adversary's objective is c times the Victim's mean reward accumulated over training.

4.3. Zero-Shot Test-Time Manipulation

In zero-shot test-time manipulation, the Adversary attempts to maximise its objective $\mathcal{J}$ during some notion of test-time starting at time I. In practice, we introduce a separate Adversary for test-time, parameterized by its own set of parameters ψ. However, both the train-time Adversary ϕ and test-time Adversary ψ have identical objective function $\mathcal{J}$. The train-time Adversary wants to *create* a backdoor to make the Victim susceptible to manipulation at test-time. The test-time Adversary wants to *use* this backdoor to control the Victim. The test-time Adversary ψ operates zero-shot because it has not seen the specific, trained test-time parameters of the Victim θ' of the current meta-episode before interacting with it.

Theoretical Results For the theoretical results, we point the reader to Appendix B.1 and B.2. Moreover, in Appendix B.3, we informally show that removing any component from a Cheap Talk MDP would either nullify all possibility of influence or make the setting so limited as to be uninteresting.

5. Experiments and Results

We evaluate ACT on three different environments: Cartpole, Pendulum, and Reacher (Brockman et al., 2016). The Victim is trained with Proximal Policy Optimisation (Schulman et al., 2017, PPO), a state-of-the-art RL algorithm. The Adversary is trained using ES (Salimans et al., 2017).

Training details are provided in Appendix E. Note that the PPO implementation uses observation normalisation, so each dimension of the observation has a mean of zero and a standard deviation of one. We also include videos of the Victim's performance and visualizations of the Adversary's outputs in the Supplementary Materials for each of the scenarios below.

5.1. Train-Time Influence

Figure 1 show the results of training Victims alongside different Adversaries. It is evaluated on four different Adversaries:

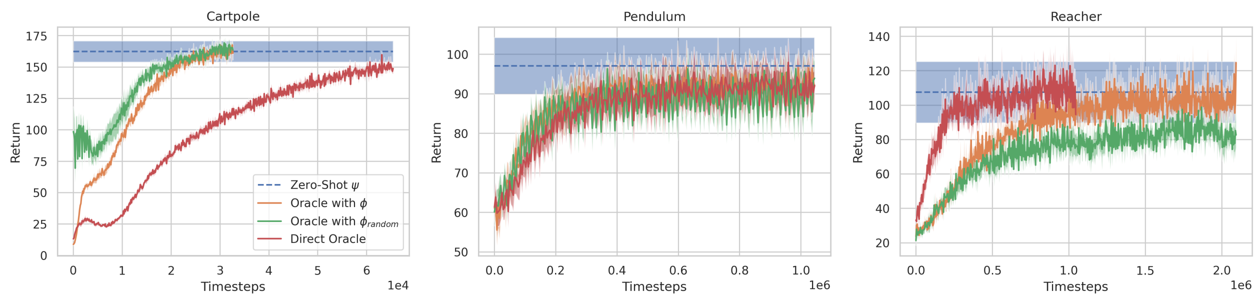
1. **Ally**: meta-trained to *maximize* the Victim's mean reward throughout training.
2. **Adversary**: meta-trained to *minimize* the Victim's mean reward throughout training.
3. **Random Adversary**: randomly initialise and fix the Adversary's parameters ϕ.
4. **Zeroes Adversary**: appends only zeroes as messages.

Ally. The Ally manages to assist the Victim to learn and converge faster – this is likely done by appending useful features of the environment. Interestingly, in Figure 5b Appendix C, we show that this Adversary even outperforms an Oracle that outputs the optimal policy logits as messages.

We hypothesise that the Adversary may be inducing catastrophic interference within the environment, which was observed by Fedus et al. (2020) in Atari 2600 games. In Figure 6 Appendix C, we demonstrate that the Adversary induces catastrophic interference in both the Adversarial setting by influencing the correlation between gradient updates between different parts of a single inner loop episode. We also study how the cheap-talk channel size affects the performance of the Adversary in Figure 5a Appendix C.

5.2. Zero-Shot Test-Time Manipulation

In the setting of zero-shot test-time manipulation, the Adversary's objective is to maximize the score of a *goal-conditioned objective*. As a consequence, the Adversary needs to learn to introduce a backdoor during train-time and use the backdoor during test-time to fully control the



(a) (b) (c)

Figure 2: Training curves of the different agents in (a) Goal-Conditioned Cartpole (b) Goal-Conditioned Pendulum (c) Goal-Conditioned Reacher. The ablations show that the train- and test-time Adversarieslearn near-optimal performance in comparison to the oracles. Error bars denote the standard error across 10 seeds of Victim trained against a single trained Adversary.

Victim. We describe the environment-specific rewards and how these goals are parameterized in Figure 3, Appendix C.

To better understand the capability of our model, we investigate four different Adversary-Victim settings. These four settings serve as ablations to study the individual and joint performance of the train- and test-time Adversaries.

1. **Direct Oracle:** In this baseline, there is no cheap talk. We simply train a PPO agent to maximize the *goal-conditioned return*. It can observe the full state and directly output actions in the environment.

2. **Zero-Shot Adversary:** First, we train a Victim θ alongside a train-time Adversary ϕ. We then evaluate the return of the test-time Adversary ψ according to the goal-conditioned return (as described in Algorithm 2). The test-time Adversary ψ operates zero-shot, because it was not trained with the specific, trained instance of the Victim θ before interacting with it. It is thus represented by a horizontal line in Figure 2.

3. **Oracle with Learned Adversary:** First, we optimize the Victim θ by training it alongside our train-time Adversary ϕ. Then, instead of ES, we use PPO to train the test-time Adversary ψ^* against the Victim θ. Unlike the zero-shot Adversary, the oracle ψ^* is allowed to train against the pretrained and fixed Victim θ to maximize its returns, as described in Algorithm 4 in Appendix D.

4. **Oracle with Random Adversary:** First, we obtain a Victim θ by training it alongside a *random* train-time Adversary, ϕ_{random}, with randomly initialized and fixed parameters. Next, we use PPO to train the test-time Adversary ψ^* to maximize the goal-conditioned return.

All results are shown in Figure 2. We can use the Direct Oracle as a baseline to measure how effective the train-time Adversary ϕ and test-time Adversary ψ are at achieving the maximal possible return jointly. As Figure 2 shows, the ES optimized train- and test-time Adversariesperform near-optimally. We investigate this further in Figure 4, Appendix C, where we compare the range and variance of Victims trained with ES optimized Adversariesϕ and Victims trained with random Adversaries ϕ_{random} across different message values.

6. Conclusion & Future Work

In this paper, we propose a novel, minimum-viable, adversarial setting for RL agents, where the Adversary can only influence the Victim over messages, and can only do so with deterministic function that only depends on the current state.

By training a Adversary with adversarial cheap talk (ACT), we show that appending to the observations of a learning agent, even with strong constraints, is sufficient to drastically improve or *decrease* a learning agent's train-time performance or introduce a backdoor to control the learning agent at test time completely. Our test-time ablation studies demonstrate that the train- and test-time Adversaries achieve near-optimal performance individually as well as jointly, when compared against strong oracle baselines. We also provide in-depth analysis on how the Adversaries work.

As RL models become more widespread, we believe practitioners must consider this new class of minimum viable attacks. Therefore, we propose identifying and filtering out seemingly-superfluous information as the first defence measure. In future work we will investigate different defence strategies, such as the identification of messages, and larger-scale input settings.

References

C. Ashcraft and K. Karra. Poisoning deep reinforcement learning agents with in-distribution triggers. *arXiv preprint arXiv:2106.07798*, 2021.

E. Bengio, J. Pineau, and D. Precup. Interference and generalization in temporal difference learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 767–777, 2020.

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

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

K. Cao, A. Lazaridou, M. Lanctot, J. Z. Leibo, K. Tuyls, and S. Clark. Emergent communication through negotiation. In *6th International Conference on Learning Representations*, 2018.

V. P. Crawford and J. Sobel. Strategic information transmission. *Econometrica*, 50(6):1431–1451, 1982.

J. Farrell. Cheap talk, coordination, and entry. *The RAND Journal of Economics*, 18(1):34–39, 1987.

W. Fedus, D. Ghosh, J. D. Martin, M. G. Bellemare, Y. Bengio, and H. Larochelle. On catastrophic interference in atari 2600 games. *arXiv preprint arXiv:2002.12499*, 2020.

J. Foerster, R. Y. Chen, M. Al-Shedivat, S. Whiteson, P. Abbeel, and I. Mordatch. Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 122–130, 2018.

J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 29, pages 2137–2145, 2016.

A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, and S. Russell. Adversarial policies: Attacking deep reinforcement learning. In *8th International Conference on Learning Representations*, 2020.

T. Gu, B. Dolan-Gavitt, and S. Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

S. H. Huang, N. Papernot, I. J. Goodfellow, Y. Duan, and P. Abbeel. Adversarial attacks on neural network policies. In *5th International Conference on Learning Representations, Workshop Track Proceedings*, 2017.

P. Kiourti, K. Wardega, S. Jha, and W. Li. Trojdrl: Evaluation of backdoor attacks on deep reinforcement learning. In *57th ACM/IEEE Design Automation Conference*, pages 1–6, 2020.

J. Kos and D. Song. Delving into adversarial attacks on deep policies. In *5th International Conference on Learning Representations, Workshop Track Proceedings*, 2017.

R. T. Lange. evosax: Jax-based evolution strategies, 2022.

D. Lenton, F. Pardo, F. Falck, S. James, and R. Clark. Ivy: Templated deep learning for inter-framework portability. *arXiv preprint arXiv:2102.02886*, 2021.

A. Letcher, D. Balduzzi, S. Racanière, J. Martens, J. N. Foerster, K. Tuyls, and T. Graepel. Differentiable game mechanics. *J. Mach. Learn. Res.*, 20:84:1–84:40, 2019a.

A. Letcher, J. N. Foerster, D. Balduzzi, T. Rocktäschel, and S. Whiteson. Stable opponent shaping in differentiable games. In *7th International Conference on Learning Representations*, 2019b.

C. Lu, T. Willi, C. Schroeder de Witt, and J. Foerster. Model-free opponent shaping. *arXiv preprint arXiv:2205.01447*, 2022.

C. Lyle, M. Rowland, and W. Dabney. Understanding and preventing capacity loss in reinforcement learning. *arXiv preprint arXiv:2204.09560*, 2022.

A. Salem, R. Wen, M. Backes, S. Ma, and Y. Zhang. Dynamic backdoor attacks against machine learning models. *arXiv preprint arXiv:2003.03675*, 2020.

T. Salimans, J. Ho, X. Chen, and I. Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

X. Song, Y. Jiang, S. Tu, Y. Du, and B. Neyshabur. Observational overfitting in reinforcement learning. In *8th International Conference on Learning Representations*, 2020.

H. van Hasselt, Y. Doron, F. Strub, M. Hessel, N. Sonnerat, and J. Modayil. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

L. Wang, Z. Javed, X. Wu, W. Guo, X. Xing, and D. Song. BACKDOORL: backdoor attack against competitive reinforcement learning. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 3699–3705, 2021.

T. Willi, J. Treutlein, A. Letcher, and J. Foerster. COLA: consistent learning with opponent-learning awareness. *arXiv preprint arXiv:2203.04098*, 2022.

A. Related Work

A.1. Test-Time Adversarial Attacks

Most work investigating adversarial attacks on deep RL systems focus on attacks at test-time, i.e. those that assume a fully trained, static policy. Gleave et al. (2020) learn adversarial policies to attack a pretrained agent at test-time. In contrast to our method, the adversarial agent can directly interact with the environment and the victim agent, thus introducing non-stationarity and assuming sampling access to the (static) victim. Also, they do not investigate adversarial agents that affect training performance. Huang et al. (2017) also investigate adversarial attacks to influence test-time performance. In contrast to our work, they directly perturb the observation space and do not simply append to it, thus assuming access to the observation space of the victim. Kos and Song (2017) also attack test-time performance by directly perturbing observations.

A.2. Backdoor Attacks

Backdoor attacks in reinforcement learning aim to introduce a vulnerability during train-time, which can be used at test-time. Backdoors can be static, meaning they get activated with fixed patterns, or dynamic, which is when the backdoor gets activated by context-dependent patterns (Salem et al., 2020). For static backdoors, the adversary often directly perturbs the observation space (Gu et al., 2017; Kiourti et al., 2020; Ashcraft and Karra, 2021). To introduce dynamic backdoors, the threat model assumes that the adversary has full control over the training process of the agent, giving the adversary the ability to introduce backdoors at train-time (Wang et al., 2021). In contrast, in our threat model, we assume a minimal range of influence by only appending to the observations. Furthermore, instead of perturbing the observations directly, Wang et al. (2021) deploy the adversarial agent directly in the environment. Interacting with the environment directly allows the adversary to introduce non-stationarity and stochasticity. In contrast, our setting does not allow the Adversary to introduce either.

A.3. Failure Modes in Deep Reinforcement Learning

Previous works have shown that using neural networks as function approximators in reinforcement learning often results in multiple failure modes due to the non-stationarity of value function bootstrapping (van Hasselt et al., 2018). In particular, works have shown that catastrophic interference (Bengio et al., 2020) and capacity loss (Lyle et al., 2022) often occur, even within a single episode of an environment (Fedus et al., 2020). Song et al. (2020) shows that deep reinforcement learning algorithms can often overfit to spurious correlations in the observation space. By appending to the observation space, we learn to induce the observational failure modes described in these works.

A.4. Opponent Shaping / Cheap Talk

Our method is closely related to the field of opponent shaping. Originally, most opponent shaping algorithms assumed white-box access to their opponents to shape the flow of the opponent's gradient (Foerster et al., 2018; Letcher et al., 2019a;b; Willi et al., 2022). Instead, Lu et al. (2022) introduce a method to shape opponents without white-box access. However, they still deploy an agent to interact directly in the environment. In contrast, we propose a method to shape other agents without having to interact in the environment at all, solely by appending messages through a cheap talk channel. Cheap talk is communication that incurs no cost, is non-binding (it can be ignored and does not limit the agent's action space), and is unverifiable (meaning any information, true or false, can be communicated) (Farrell, 1987). In RL terms, a cheap talk channel is a part of the state space which can be observed by other agents but does not alter transition dynamics or reward functions. Cheap talk channels (Crawford and Sobel, 1982) in deep reinforcement learning have been used to learn emergent communication (Foerster et al., 2016) and to solve coordination problems (Cao et al., 2018). To the best of our knowledge, this paper is the first to use a cheap talk channel (and only a cheap talk channel) to shape learning agents.

B. Proofs

B.1. Proof of Proposition 1

In this section, we further justify the claim that our setting represents the bare minimum range of influence. To begin, we prove that Adversaries *cannot* influence *tabular* Victims in Cheap Talk MDPs; Adversaries can therefore only interfere with a Victim through their function approximator.

Proposition 1. *For any deterministic Adversary $f : \mathcal{S} \to \mathcal{M}$, the return of a tabular Victim initialised uniformly along the*

$\mathcal{M}$ axis is independent from f. Moreover, any Victim which is guaranteed to converge to optimal policies in MDPs will, for any Cheap Talk MDP, converge to a policy whose expected return is the optimal return for the original no-channel MDP – even in non-tabular settings and regardless of initialisation.

Proof. We begin with the tabular case.

Tabular Victims. In a Cheap Talk MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mathcal{M}, f, \mathcal{J} \rangle$, a tabular Victim arbitrarily orders states as $\{s_1, \ldots, s_d\}$ and messages as $\{m_1, \ldots, m_k\}$, where $d = |\mathcal{S}|$ and $k = |\mathcal{M}|$, and stores policies $\pi_t(\cdot \mid s_i, m_j)$ at time t of the learning process for all $i \in [d], j \in [k]$. The argument follows identically for value functions. Assuming uniform initialisation along the $\mathcal{M}$ axis means that

$$\pi_0(\cdot \mid s_i, m_j) = \pi_0(\cdot \mid s_i, m_{j'})$$

for all $j, j' \in [k]$. Now consider any two Adversaries f, g and their influence on two copies of the same Victim V, W with respective policies π, χ. The only states encountered in the environment are of the form $(s, f(s))$ and $(s, g(s))$ respectively, so Victims only update the corresponding policies

$$\pi_t(\cdot \mid s_i, f(s_i)) \qquad \text{and} \qquad \chi_t(\cdot \mid s_i, g(s_i)) \, .$$

We prove by induction that these quantities are equal for all t. The base case holds by uniform initialisation along $\mathcal{M}$; assume the claim holds for all fixed $0 \leq t \leq T$. The Victims update their policies at time $T+1$ according to the same learning rule, as a function of the transitions and returns under current and past policies π_t and χ_t respectively. Transitions take the form $(s, f(s), a, s', f(s'))$ for V and $(s, g(s), a, s', g(s))$ for W, which have identical probabilities and returns because

$$\pi_t(a \mid s_i, f(s_i)) = \chi_t(a \mid s_i, g(s_i)) \, ;$$
$$\mathcal{P}(s', f(s') \mid s, f(s), a) = \mathcal{P}(s', g(s') \mid s, g(s), a) \, ;$$
$$\mathcal{R}(s, f(s), a) = \mathcal{R}(s, g(s), a)$$

by inductive assumption and independence of $\mathcal{P}, \mathcal{R}$ from $\mathcal{M}$. This probability- and return-preserving bijection between transitions, as well as being copies with identical initialisation in the environment, implies that policies $\pi_T(\cdot \mid s_i, f(s_i)) = \chi_T(\cdot \mid s_i, g(s_i))$ are updated identically to

$$\pi_{T+1}(\cdot \mid s_i, f(s_i)) = \chi_{T+1}(\cdot \mid s_i, g(s_i))$$

as required to complete induction. Note that this could not necessarily be accomplished in non-tabular settings, where updating parameters θ of the function approximator for some state s_i may alter the policy on some other state s_j. It now follows that trajectories $\tau = (s^k, f(s^k), a^k)_k$ for V and $\omega = (s^k, g(s^k), a^k)_k$ for W have identical probabilities and hence produce identical returns

$$\mathbb{E}_{\tau \sim \pi_t}\left[\mathcal{R}(\tau)\right] = \mathbb{E}_{\omega \sim \chi_t}\left[\mathcal{R}(\omega)\right]$$

at any timestep t of the learning process, concluding independence from Adversaries.

Optimally Convergent Victims. By assumption, the Victim is guaranteed to converge to an optimal policy $\bar{\pi}$ in the Cheap Talk MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{M}, f, \mathcal{J}, \gamma \rangle$, since a Cheap Talk MDP is itself an MDP with an augmented state space $\mathcal{S} \times \mathcal{M}$ and augmented transition/reward functions that are defined to be independent from $\mathcal{M}$. Now $\bar{\pi}$ naturally induces a policy π on the no-channel MDP, given by $\pi(\cdot \mid s) \coloneqq \bar{\pi}(\cdot \mid s, f(s))$, and in particular $Q(s,a) = \bar{Q}(s, f(s), a)$ by independence of transitions and rewards from $\mathcal{M}$. Optimality of π follows directly from the Bellman equation

$$\begin{aligned} Q(s,a) = \bar{Q}(s, f(s), a) &= \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s,a), r \sim \mathcal{R}(\cdot|s,a)}\left[r + \gamma \max_{a' \in \mathcal{A}} \bar{Q}(s', f(s'), a')\right] \\ &= \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s,a), r \sim \mathcal{R}(\cdot|s,a)}\left[r + \gamma \max_{a' \in \mathcal{A}} Q(s', a')\right] \, . \end{aligned}$$

Now trajectories $\bar{\tau} = (s^k, f(s^k), a^k)_k$ and $\tau = (s^k, a^k)_k$ have identical probability and return under π and $\bar{\pi}$ respectively, so the Victim has expected return

$$\mathbb{E}_{\bar{\tau} \sim \bar{\pi}}\left[\mathcal{R}(\bar{\tau})\right] = \mathbb{E}_{\tau \sim \pi}\left[\mathcal{R}(\tau)\right]$$

which is the optimal expected return of the original no-channel MDP. □

B.2. Proof of Proposition 2

For completeness, we also formally prove our claims from the introduction regarding what the Adversary *cannot* do in Cheap Talk MDPs.

Proposition 2. *In a Cheap Talk MDP, the Adversary cannot (1) occlude the ground truth, (2) influence the environment dynamics / reward functions, (3) see the Victim's actions or parameters, (4) inject stochasticity, or (5) introduce non-stationarity.*

Proof. Mostly by definition. Formally, consider a Cheap Talk MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mathcal{M}, f, \mathcal{J} \rangle$ as defined in Section 3. For a fixed training / testing run of the Victim on the MDP, the Adversary outputs a message $f(s)$ at each step according to a fixed deterministic function $f : \mathcal{S} \to \mathcal{M}$.

(1) The message is *appended* to the state s and the Victim acts with full visibility of the ground truth (state) s according to its policy: $a \sim \pi(\cdot \mid s, f(s))$.

(2) The transition and reward functions $\mathcal{P}, \mathcal{R}$ are defined to be independent from $\mathcal{M}$. Formally we have $\mathcal{P}(\cdot \mid s, m, a) = \mathcal{P}(\cdot \mid s, m', a)$ for all $m, m' \in \mathcal{M}$ (similarly for $\mathcal{R}$), so the Adversary's choice of message $m = f(s)$ cannot influence $\mathcal{P}$ or $\mathcal{R}$.

(3) $f : \mathcal{S} \to \mathcal{M}$ is defined as a function of $\mathcal{S}$ only, so the Adversary cannot condition its policy based on the Victim's actions or parameters (i.e. it cannot see them for all practical purposes).

(4) f is a deterministic function, so $\pi(\cdot \mid s, f(s))$ is a distribution only on actions $\mathcal{A}$. The transition and reward functions are independent from f, so they are distributions only on state-action pairs $\mathcal{S} \times \mathcal{A}$. It follows that the Adversary injects no further stochasticity into the MDP.

(5) f is static for a fixed training / testing run, so $s_t = s_{t'}$ implies $f(s_t) = f(s_{t'})$ for all timesteps t, t' in the run. It follows that any given Victim policy π is stationary, namely $\pi(\cdot \mid s_t, f(s_t)) = \pi(\cdot \mid s_{t'}, f(s_{t'}))$ for all $s_t = s_{t'}$. Since $\mathcal{P}$ and $\mathcal{R}$ are stationary (as defined by a standard MDP) and independent from $\mathcal{M}$, their stationarity is also preserved. □

B.3. Informal Justification of Minimality

Finally, let us informally show that removing any component from a Cheap Talk MDP would either nullify all possibility of influence or make the setting so limited as to be uninteresting.

(1) Removing the set $\mathcal{M}$ or the policy $f : \mathcal{S} \to \mathcal{M}$ entirely would result in the Victim being completely independent from the Adversary, since nothing would be appended to its observation.

(2) Restricting the capacity of $\mathcal{M}$ to a certain number of bits would further restrict an Adversary's range of influence, so one could say that the *truly* minimum-viable setting is to impose a set of size $|\mathcal{M}| = 1$. However, cheap talk is still cheap talk when varying capacity, and there is no reason to arbitrarily restrict the size to 1 if we are to apply our setting to complex environments likely requiring more than a single bit of communication to witness interesting results.

(3) Not allowing Adversaries to see states, namely removing $\mathcal{S}$ as inputs to f, yields a function $f : \{0\} \to \mathcal{M}$ which always outputs the same message$f(0) = m \in \mathcal{M}$. This is equivalent to the previous restriction of imposing a set $\mathcal{M}$ of size 1, since in this case any function $f : \mathcal{S} \to \mathcal{M}$ would have to output the unique element $f(s) = m$ for all input states s.

(4) The Adversary must have some objective function $\mathcal{J}$ in order for an adversarial setting to make sense – removing it would remove the Adversary's rationale for existence, since it would have no incentive to learn parameters that influence the Victim according to some goal.

(5) Restricting the function class of objectives $\mathcal{J}$ is a valid minimisation of the setting, but simply restricts our interesting the setting itself. The setting should at the very least allow for adversarial objectives of the form $\mathcal{J} = -J$ as we consider in the train-time setting. In test-time, our aim is to show how Adversaries can exert arbitrary control over Victims despite cheap talk restrictions, and we therefore consider more general objective functions.

C. Experiments and Results Additional Material

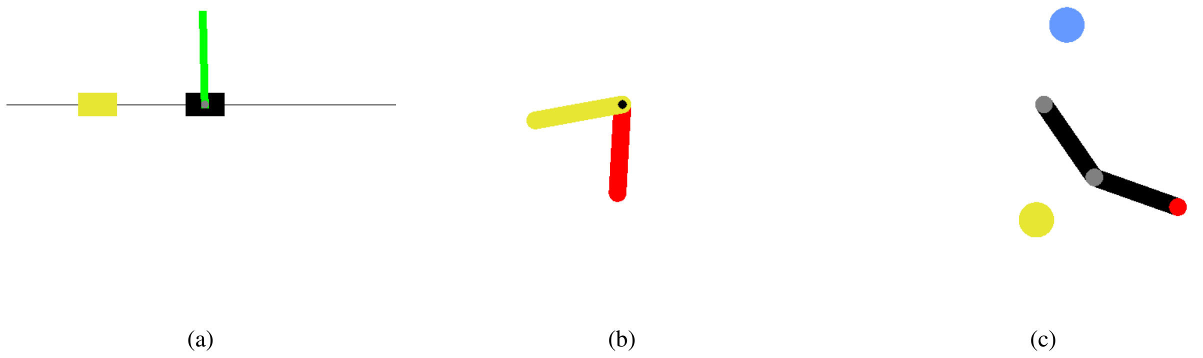
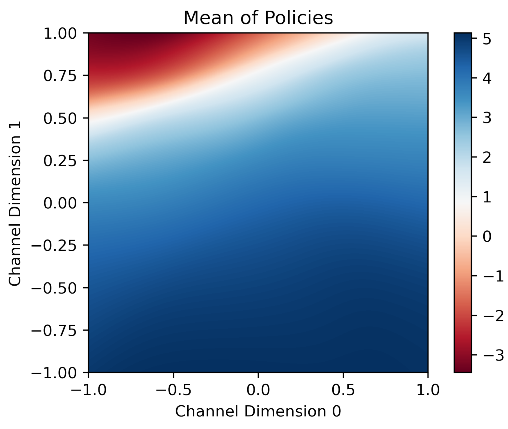
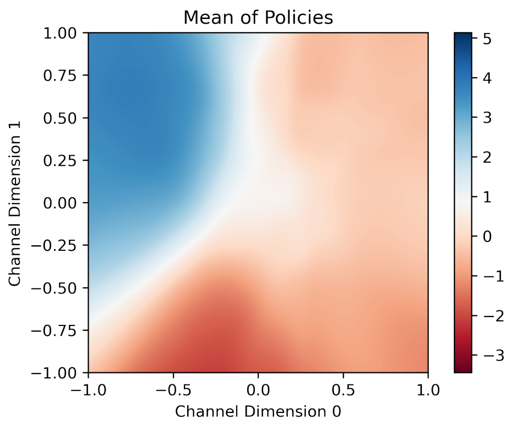


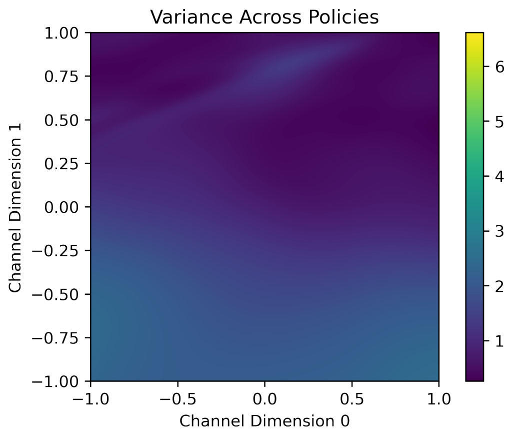
Figure 3: Visualizations of our goal-conditioned environments (a) In Cartpole, the Adversary's target is a randomly selected point on the x-axis, indicated by the yellow box. (b) In Pendulum, the Adversary's goal is a randomly selected angle indicated by the yellow pole. (c) In Goal-Conditioned Reacher, the Adversary's goal is a specific point, denoted by the yellow circle, while the Victim's goal is the blue circle.



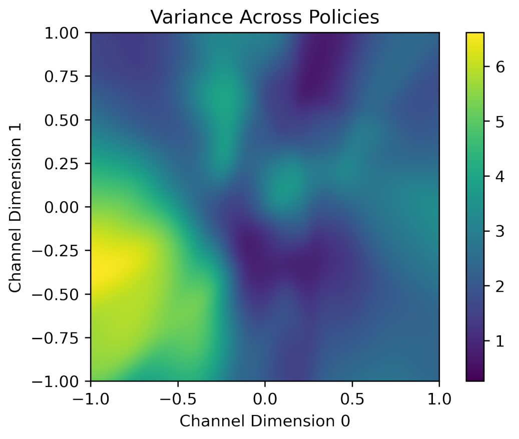
(a)



(b)

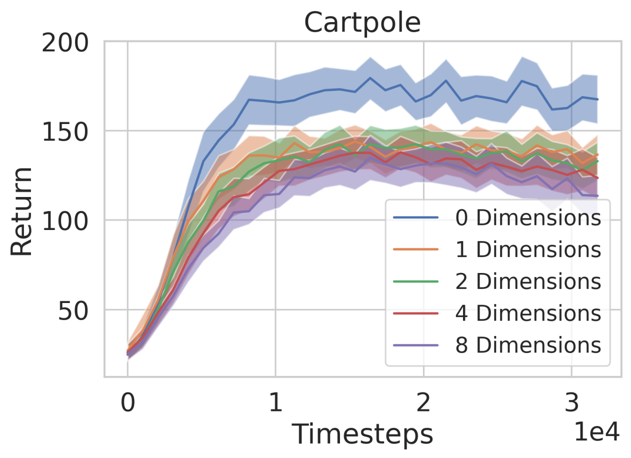


(c)

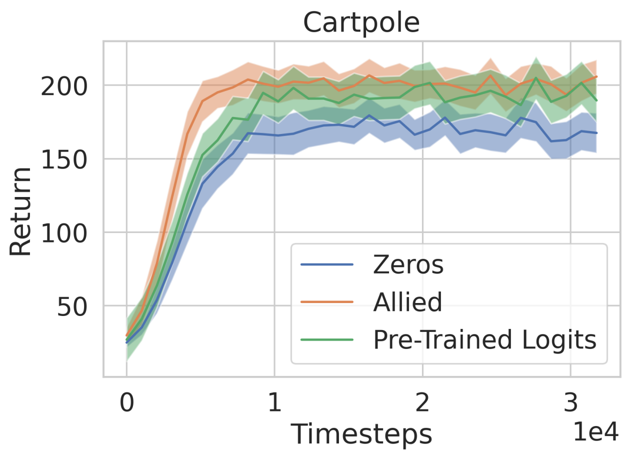


(d)

Figure 4: We train 10 different Victims alongside the Learned ϕ (a & c), as well as 10 different Victims alongside a randomly generated ϕ (b & d) in the Pendulum environment. (a) and (b) show the mean of the policy output across the 10 Victims as we vary the value of the message in a fixed randomly selected state. Notably, the policies trained with the learned ϕ achieve a much wider range of outputs. (c) and (d) show the variance of the policy output across the 10 Victims. Notably, the policies trained with the learned ϕ display very little variance, implying that the learned ϕ shapes the Victim in a consistent way.

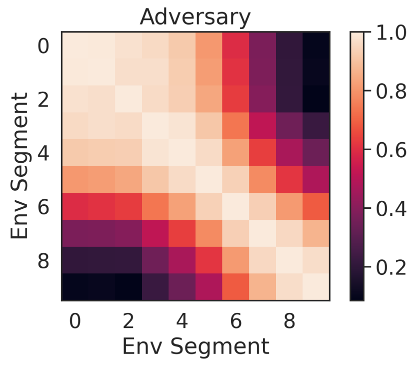


(a)

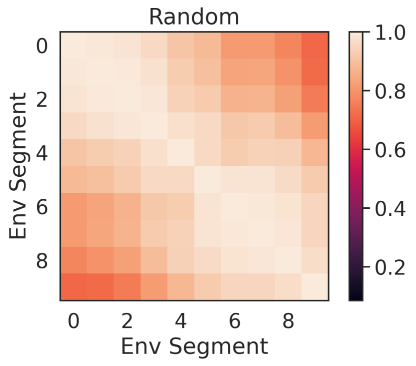


(b)

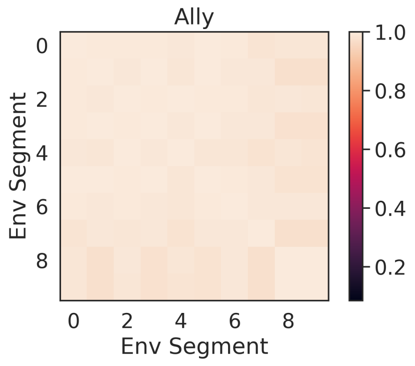
Figure 5: (a) Ablations on the different number of cheap talk dimensions for the Adversary in Cartpole (b) Comparing the ally with an Adversary that outputs the optimal logits in Cartpole. Error bars denote the standard error across 10 seeds of a Victim trained against a single meta-trained Adversary.



(a)



(b)



(c)

Figure 6: To perform this analysis, we collect each Victim's experience buffer, before the agents have converged in training, and split each one into 10 bins, ordered by the time-step within the environment. We then calculate the gradient update the agents would perform on each of these bins. In the Adversarial setting (a), the gradient updates performed for transitions sampled early in an episode can *interfere* with the gradient updates performed for transitions later in an episode. Meanwhile, in the Allied setting (c), those gradient updates are positively correlated, suggesting that the gradient updates aid each other.

D. Pseudocode

Algorithm 1 Train-time ACT

```
1: Set $c = \pm 1$ for allied / adversarial
2: Initialize Adversary parameters $\phi$
3: for $m = 0$ to $M$ do
4:     Sample $\phi_n \sim \phi + \sigma\epsilon_n$ where $\epsilon_1, ..., \epsilon_N \sim \mathcal{N}(0, I)$
5:     for $n = 0$ to $N$ do
6:         Initialize Victim parameters $\theta$
7:         rewards = []
8:         for $e = 0$ to $E$ do
9:             s = env.reset()
10:            while not done do
11:                $\bar{s} = [s, f_{\phi_n}(s)]$
12:                $a \sim \pi_\theta(\cdot \mid \bar{s})$
13:                $r, s$, done = env.step($a$)
14:                rewards.append($r$)
15:            end while
16:            Update $\theta$ with PPO to maximise $J$
17:        end for
18:        $\mathcal{J}_n = c \cdot$ sum(rewards)/len(rewards)
19:    end for
20:    Update $\phi$ using ES to maximise $\mathcal{J}$
21: end for
```

Algorithm 2 Test-time ACT

```
 1: Initialize train-time ACT parameters $\phi$
 2: Initialize test-time ACT parameters $\psi$
 3: for $m = 0$ to $M$ do
 4:     Sample $\phi_n \sim \phi + \sigma\epsilon_n$ where $\epsilon_1, ..., \epsilon_N \sim \mathcal{N}(0, I)$
 5:     Sample $\psi_n \sim \psi + \sigma\epsilon_n$ where $\epsilon_1, ..., \epsilon_N \sim \mathcal{N}(0, I)$
 6:     for $n = 0$ to $N$ do
 7:         Initialize policy params $\theta$
 8:         rewards = []
 9:         for $e = 0$ to $E$ do
10:             s = env.reset()
11:             while not done do
12:                 $m = f_{\phi_n}(s)$
13:                 $\bar{s}$ = [$s$, $m$]
14:                 $a \sim \pi_\theta(\cdot \mid \bar{s})$
15:                 $r, s$ = env.step($a$)
16:             end while
17:             Update $\theta$ using PPO to maximise $J$
18:         end for
19:         for $i = 0$ to $I$ do
20:             s = env.reset()
21:             while not done do
22:                 $m = f_{\psi_n}(s)$
23:                 $\bar{s}$ = [$s$, $m$]
24:                 $a \sim \pi_\theta(\cdot \mid \bar{s})$
25:                 $r$, $s$, done = env.step($a$)
26:                 $r_t^S = R^S(s, a)$
27:                 rewards.append($r_t^S$)
28:             end while
29:         end for
30:     end for
31:     Update $\phi$ using ES to maximise $\mathcal{J}$
32:     Update $\psi$ using ES to maximise $\mathcal{J}$
33: end for
```

Algorithm 3 Test-time Oracle PPO ACT

```
1:  Initialize train-time ACT parameters $\phi$
2:  Obtain trained $\phi$, $\theta$ from Algorithm 2
3:  Initialize test-time ACT parameters $\psi^*$
4:  **for** $i = 0$ **to** $I$ **do**
5:      s = env.reset()
6:      **while** not done **do**
7:          $m \sim \pi_{\psi^*}(\cdot \mid s)$
8:          $\bar{s}$ = $[s, m]$
9:          $a \sim \pi_\theta(\cdot \mid \bar{s})$
10:         $r, s$, done = env.step($a$)
11:         $r_t^S = R^S(s, a)$
12:         rewards.append($r_t^S$)
13:     **end while**
14:     Update $\psi^*$ using PPO to maximise $\mathcal{J}$
15: **end for**
```

Algorithm 4 Test-time Random Shaper

```
1: Initialize train-time ACT parameters $\phi_{\text{random}}$
2: Initialize policy params $\theta$
3: rewards = []
4: **for** $e = 0$ **to** $E$ **do**
5:     s = env.reset()
6:     **while** not done **do**
7:         $m = f_{\phi_{\text{random}}}(s)$
8:         $\bar{s}$ = $[s, m]$
9:         $a \sim \pi_\theta(\cdot \mid \bar{s})$
10:        $r, s$ = env.step($a$)
11:    **end while**
12:    Update $\theta$ using PPO to maximise $J$
13: **end for**
14: Initialize test-time ACT parameters $\psi^*$
15: **for** $i = 0$ **to** $I$ **do**
16:    s = env.reset()
17:    **while** not done **do**
18:        $m \sim \pi_{\psi^*}(\cdot \mid s)$
19:        $\bar{s}$ = $[s, m]$
20:        $a \sim \pi_\theta(\cdot \mid \bar{s})$
21:        $r, s$ = env.step($a$)
22:        $r_t^S = R^S(s, a)$
23:        rewards.append($r_t^S$)
24:    **end while**
25:    Update $\psi^*$ using PPO to maximise $\mathcal{J}$
26: **end for**
```

E. Hyperparameter Details

We train thousands of agents per minute on a single V100 GPU by vectorising both the *PPO algorithm itself* and the environments using Jax (Bradbury et al., 2018). This allows us to JIT-compile the *full training pipeline* and perform end-to-end deep RL training completely on GPUs. We adapt the environment implementations from Brockman et al. (2016) and Lenton et al. (2021) and use the ES implementation from Lange (2022). This compute setup allows us to efficiently perform outer-loop ES on the full training trajectories of inner-loop PPO agents. For example, in Cartpole, we run 8192 PPO Victims alongside 8192 train-time Adversariesand 8192 test-time Adversaries, each over four instances of the environment on a single V100 GPU. Over 1024 generations of ES, this results in training 8,388,608 PPO agents from scratch in 2 hours on 4 V100 GPUs.

We report the hyperparameter values used for each environment in our experiments.

Table 1: Important parameters for the Cartpole environment

Parameter	Value
State Size	4
message Size	2
Number of Environments	4
Maximum Grad Norm	0.5
Number of Updates	32
Update Period	256
Outer Discount Factor γ	0.99
Number of Epochs per Update	16
PPO Clipping ϵ	0.2
General Advantage Estimation λ	0.95
Critic Coefficient	0.5
Entropy Coefficient	0.01
Learning Rate	0.005
Population Size	1024
Number of Generations	2049
Outer Agent (OA) Hidden Layers	2
OA Size of Hidden Layers	64
OA Hidden Activation Function	ReLU
OA Output Activation Function	Tanh
Inner Agent (IA) Actor Hidden Layers	2
IA Size of Actor Hidden Layers	32
IA Number of Critic Hidden Layers	2
IA Size of Critic Hidden Layers	32
IA Activation Function	Tanh
Number of Rollouts	4

Table 2: Important parameters for the Pendulum environment

Parameter	Value
State Size	3
message Size	2
Number of Environments	16
Maximum Grad Norm	0.5
Number of Updates	128
Update Period	256
Outer Discount Factor γ	0.95
Number of Epochs per Update	16
PPO Clipping ϵ	0.2
General Advantage Estimation λ	0.95
Critic Coefficient	0.5
Entropy Coefficient	0.005
Learning Rate	0.02
Population Size	768
Number of Generations	2049
Outer Agent (OA) Hidden Layers	2
OA Size of Hidden Layers	64
OA Hidden Activation Function	ReLU
OA Output Activation Function	Tanh
Inner Agent (IA) Actor Hidden Layers	1
IA Size of Actor Hidden Layers	32
IA Number of Critic Hidden Layers	1
IA Size of Critic Hidden Layers	32
IA Activation Function	Tanh
Number of Rollouts	4

Table 3: Important parameters for the Reacher environment

Parameter	Value
State Size	10
message Size	4
Number of Environments	32
Maximum Grad Norm	0.5
Number of Updates	256
Update Period	128
Outer Discount Factor γ	0.99
Number of Epochs per Update	10
PPO Clipping ϵ	0.2
General Advantage Estimation λ	0.95
Critic Coefficient	0.5
Entropy Coefficient	0.0005
Learning Rate	0.004
Population Size	128
Number of Generations	2049
Outer Agent (OA) Hidden Layers	2
OA Size of Hidden Layers	64
OA Hidden Activation Function	ReLU
OA Output Activation Function	Tanh
Inner Agent (IA) Actor Hidden Layers	2
IA Size of Actor Hidden Layers	128
IA Number of Critic Hidden Layers	2
IA Size of Critic Hidden Layers	128
IA Activation Function	ReLU
Number of Rollouts	4