ARC: Adversarially Robust Control Policies for Autonomous Vehicles

Sampo Kuutti, Saber Fallah, Richard Bowden

Abstract— **Deep neural networks have demonstrated their capability to learn control policies for a variety of tasks. However, these neural network-based policies have been shown to be susceptible to exploitation by adversarial agents. Therefore, there is a need to develop techniques to learn control policies that are robust against adversaries. We introduce Adversarially Robust Control (ARC), which trains the protagonist policy and the adversarial policy end-to-end on the same loss. The aim of the protagonist is to maximise this loss, whilst the adversary is attempting to minimise it. We demonstrate the proposed ARC training in a highway driving scenario, where the protagonist controls the follower vehicle whilst the adversary controls the lead vehicle. By training the protagonist against an ensemble of adversaries, it learns a significantly more robust control policy, which generalises to a variety of adversarial strategies. The approach is shown to reduce the amount of collisions against new adversaries by up to 90.25%, compared to the original policy. Moreover, by utilising an auxiliary distillation loss, we show that the fine-tuned control policy shows no drop in performance across its original training distribution.**

I. Introduction

The powerful function approximation capabilities of Deep Neural Networks (DNNs) has pushed the state-of-the-art forward in multiple fields. This has lead to machine learning being adopted to learn control policies in applications such as robotic arm manipulation [1], [2], navigation [3], [4], and autonomous driving [5], [6]. In recent years, there have been numerous DNN-driven approaches proposed for autonomous vehicle control, and among them Imitation Learning has attracted attention due to its ability to learn driving behaviours from human demonstration [7]. Imitation learning performs well in naturalistic driving and scales well to training data, but performs poorly when experiencing scenarios outside of the training distribution [8], [9]. Furthermore, these learned policies have been proven to be susceptible to attacks by adversarial agents [10], [11]. These limitations pose a challenge to adapting these learned control policies to safety-critical systems.

We propose an adversarial learning framework, which uses imitation learning as a first training step and then improves the robustness to distribution shift by training the policy simultaneously against an ensemble of adversarial agents whose goal is to degrade the performance of the target policy. Both networks learn through a semi-competitive game, where one aims to drive in a safe manner and the other aims to create scenarios in which collisions could occur. Therefore, over time the target agent learns to avoid mistakes which an adversary could exploit. Our tests show that the approach maintains a safe behaviour even against learned adversarial agents, and results in a more robust and safe control policy.

This approach is partially inspired by the minimax game at the heart of Generative Adversarial Networks (GANs) [12], where two networks are trained on the same loss such that the Discriminator aims to correctly classify images as real or fake whilst the Generator aims to fool the Discriminator with generated images. GANs have also inspired the Generative Adversarial Imitation Learning (GAIL) [13], [14], where the Generator generates actions, whilst the Discriminator aims to predict whether the state-action pair comes from the Generator or the Expert. However, different from GANs or GAIL, where the Generator generates images/actions and the Discriminator performs binary classification, in our work both networks learn to predict continuous control actions for separate agents within a simulator.

Image-based DNN classifiers have been shown to be susceptible to adversarial attacks, which perturb the observations of the network, causing them to misclassify the image [15], [16]. As a common defense, adversarial training has been shown to improve robustness to adversarial attack [17]. Similarly, perturbing the observations of a policy or varying its dynamics during training in an adversarial fashion has been proven to improve the robustness of learned control policies [18]–[21]. Combining concepts of competing networks from GANs and adversarial training, Robust Adversarial Reinforcement Learning (RARL) [21]–[23] uses two DNNs trained through Reinforcement Learning (RL), where one DNN aims to learn a control policy for a given task and the other DNN aims to degrade the performance of the target policy by generating disturbances in its observations or actions. The RARL approach has also been shown to improve robustness of RL policies for different tasks, including autonomous driving. Going beyond disturbances in observation or action space, the Adversarial Policies approach by Gleave *et al.* [11] controls a separate agent in the same environment as the target policy, where the adversarial agent aims to prevent the target agent from performing in its task successfully. The Adversarial Policy was shown to learn behaviours which significantly weaken the performance of the target policy, and by fine-tuning the target policy through RL, it learned to counter the adversary. However, it was shown that new adversaries could be trained to find new weaknesses even in the fine-tuned target policy. Concurrently, several approaches have emerged in autonomous vehicle testing, where an adversarial policy is used to control an agent (e.g. vehicle, pedestrian) on the road, and aim to find behaviours

Sampo Kuutti and Saber Fallah are with the Connected and Autonomous Vehicles Lab, University of Surrey, Guildford, GU2 7XH, UK. Email: {s.j.kuutti, s.fallah}@surrey.ac.uk

Richard Bowden is with the Centre for Vision, Speech and Signal Processing, University of Surrey, GU2 7XH, UK. Email: r.bowden@surrey.ac.uk

which cause the target autonomous vehicle to make mistakes [10], [24]–[26]. This type of adversarial testing has been shown to be effective in the validation of autonomous vehicle control policies, by finding weaknesses which may not have been found through traditional validation methods [27], [28].

In this work we utilise similar adversarial agents to exploit weaknesses in the target control policy, but rather than training each agent independently, we employ a GAN-like minimax loss where the agents are trained end-to-end to compete against each other. This results in more robust control policies. We show that by taking an initially susceptible Imitation Learning vehicle motion control policy, and fine-tuning it through our ARC training framework, the policy learns to avoid collisions against the competing adversary. Moreover, we show that after adversarial fine-tuning, the resulting control policy exhibits significantly improved robustness to new adversarial agents trained against it. We also demonstrate that using an auxiliary distillation loss results in the fine-tuned control policy retaining the same level of performance across its original training distribution, thereby improving robustness to safety-critical scenarios without degrading performance in typical driving scenarios.

The remainder of this paper is as follows. Section II describes the methodology used for pre-training of the target and adversary control policies, as well as the proposed Adversarially Robust Control framework for training both networks end-to-end. The simulated experimental results are presented and discussed in Section III. Finally, concluding remarks are given in Section IV.

II. Methodology

We demonstrate our approach in a vehicle following scenario applied to highway driving. The aim of the host vehicle is to maintain a safe distance from the lead vehicle in front. To do this, the control policy infers actions which control the gas and brake pedals of the host vehicle, based on the low-dimensional states from the vehicle's radar and inertial sensors. The adversarial agent controls the lead vehicle, and is trained through Reinforcement Learning to create scenarios in which collisions are likely to occur. We first describe the training methodology for the Imitation Learning (IL) based host vehicle control policy, followed by the training of the adversarial agent. Finally, we describe our Adversarially Robust Control (ARC) formulation, where both agents are trained end-to-end through a minimax loss. We denote the Imitation Learning based agent by IL, while during the ARC training, where both networks are trained end-to-end, the Protagonist and Adversary are denoted by P and A, respectively.

A. Imitation Learning

Imitation Learning is a subset of Supervised Learning, where the model learns from expert demonstrations of trajectories [29]. Imitation learning aims to learn a control policy by imitating the behaviour of an expert, by observing states s_t^{IL} and predicting a corresponding control action a_t^{IL}, which is then compared to the expert's optimal action $\hat{a}_t$. This can be done by collecting a dataset $\mathcal{D} = \{s_t, \hat{a}_t\}_{t=0}^{N}$ of N expert demonstrations, and then training the agent to predict the expert's actions for the states in the dataset in a supervised manner. In this work, we use the Imitation Learning based vehicle motion control model from [30], which trains a feedforward neural network through Imitation Learning to predict the longitudinal control actions of a vehicle in highway driving. The Imitation Learning policy is denoted by π^{IL} and is represented by a feedforward neural network with 3 hidden layers of 50 neurons each, with the parameters θ^{IL}. Therefore, the agent's aim is to learn a policy π^{IL} which generates actions similar to the expert policy π^*, by finding the optimal parameters θ^* based on an imitation loss $\mathcal{L}^{IL}$:

$$\theta^* = \arg\min_{\theta^{IL}} \sum_t \mathcal{L}^{IL}(\pi^{IL}(s_t^{IL}|\theta^{IL}), \hat{a}_t) \tag{1}$$

The network is trained using the Mean Square Error (MSE) loss with respect to the labels given by the expert's action in dataset $\mathcal{D}$:

$$\mathcal{L}^{IL}(\mathcal{D}, \pi^{IL}(s_t^{IL}|\theta^{IL}), \hat{a}_t) = |a_t^{IL} - \hat{a}_t|^2 \tag{2}$$

The dataset was collected by driving at highway speeds on a single road, within the IPG CarMaker simulator [31]. The expert demonstrator used to collect example actions, is the default driver in the simulator, IPG Driver. The expert's aim is to maintain a 2s time headway, t_h, from the lead vehicle in front of the host vehicle. The time headway t_h is a measure of distance between two vehicles in time, as given by:

$$t_h = \frac{x_{rel}}{v} \tag{3}$$

where x_{rel} is the distance between the two vehicles in m, and v is the velocity of the host vehicle in m/s.

Each observation s_t in the dataset consists of the host vehicle velocity v, relative velocity with respect to the lead vehicle v_{rel}, and time headway t_h, such that $s_t^{IL} = (v, v_{rel}, t_h)$. The action of the agent controls the vehicle's gas and brake pedals, and is represented as a single continuous value $a_t^{IL} \in [-1, 1]$, where negative values represent the use of the brake pedal and positive values represent the use of the gas pedal.

B. Adversarial Reinforcement Learning

Reinforcement learning can be formally described by a Markov Decision Process (MDP) denoted by a tuple ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{P}$, $\mathcal{R}$, γ), where $\mathcal{S}$ is the state-space, $\mathcal{A}$ is the action-space, $\mathcal{P}$ is the transition probability model, $\mathcal{R}$ is the reward function, and γ is the discount factor. At every timestep t, the RL agent observes the state $s_t \in \mathcal{S}$ and takes an action $a_t \in \mathcal{A}$ according to its policy π. Then, the environment E transitions to the next state s_{t+1} according to the state transitions probability $p(s_{t+1}|s_t, a_t)$ as given by $\mathcal{P}$. The agent then receives a scalar reward $r_t \in \mathcal{R}$. The aim of the RL agent is to maximise its long term discounted rewards, as given by the returns R_t:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{4}$$

where the discount factor $\gamma \in [0, 1]$ is used to prioritise immediate rewards over future rewards.

To find weaknesses in the target control policy, we employ the Adversarial Testing Framework by Kuutti *et al.* [10] based on Adversarial Reinforcement Learning (ARL). The technique uses an agent trained through reinforcement learning, whose aim is to create collisions with the vehicle behind it. Therefore, this agent acts as a lead vehicle to the host vehicle control policy described in the previous subsection. However, to ensure the results are realistic and all collisions are preventable (and therefore any collisions mean the host vehicle made mistakes), the actions and states of the adversarial agents are constrained. In [10], the robustness of vehicle follower policies were tested in different velocity ranges, and the velocity range with the most collisions was $v_{lead} \in [12, 30]$ m/s. Therefore, we utilise these velocity limits for the adversary, and aim to reduce collisions by improving the robustness of the protagonist, whilst minimising any impact on the agent's behaviour in its training domain. Similarly, to ensure the collisions are avoidable, the acceleration of the lead vehicle is limited to $a_{lead} \in [-6, 2]$ m/s^2. During training of the adversarial agent, various values for the coefficient of friction in the ranges [0.4, 1.0] were used to test the response of the target agent in different driving conditions. The adversary's observations are represented by $s^A_t = (v, a, v_{rel}, t_h)$, where v is the velocity and a is the acceleration of the following vehicle. The action of the adversary a^A is a continuous value for the acceleration of the lead vehicle a_{lead}. The adversarial agent is trained through Advantage Actor Critic (A2C) [32] Reinforcement Learning, which is an actor-critic on-policy algorithm. The two networks, actor and critic networks, estimate the policy function π^A and value function $V(s^A)$. To improve training stability, the weights of both networks are updated based on the Advantage function $A(s^A, a^A)$:

$$V(s^A) = \mathbb{E}[R_t | s^A_t = s^A] \tag{5}$$

$$Q(s^A, a^A) = \mathbb{E}[R_t | s^A_t = s^A, a^A] \tag{6}$$

$$\begin{aligned} A(s^A_t, a^A_t) &= Q(s^A_t, a^A_t) - V(s^A_t) \\ &\approx \sum_{k}^{n-1} \gamma^k r_{t+k} + \gamma^n V(s^A_{t+n}) - V(s^A_t) \end{aligned} \tag{7}$$

Where $\mathbb{E}$ denotes expectation, $V(s^A_t)$ is the value function, and $Q(s^A_t, a^A_t)$ is the state-action (or quality) function [33].

To estimate the stochastic policy π^A, the actor network uses two outputs, estimated action value μ and estimated action variance σ^2. The action applied by the adversarial agent is then sampled from the Gaussian distribution $a^A_t \sim \mathcal{N}(\mu, \sigma^2)$. To do this, the actor network uses 3 hidden layers with 50 neurons, followed by a Long Short-Term Memory [34] layer with 16 units, followed by the output layer. Meanwhile, the critic network estimating the value function $V(s^A)$, uses 2 hidden layers with 50 neurons. All hidden neurons use the ReLU-6 activation, μ uses a tanh activation, σ^2 uses a softplus activation, and the value estimate uses a linear activation. To train both networks, A2C updates the actor network parameters θ^π and critic network parameters θ^V, using the policy loss $\mathcal{L}_{\pi^A}$ and value loss $\mathcal{L}_V$, respectively:

$$\mathcal{L}_v = (A(s^A_t, a^A_t))^2 \tag{8}$$

$$\mathcal{L}_{\pi^A} = -log\pi^A(a^A_t | s^A_t) A(s^A_t, a^A_t) - \beta H(\pi^A(s^A_t)) \tag{9}$$

where β is the entropy coefficient and $H(\pi^A(s^A_t))$ is the policy entropy used to encourage exploration in the adversary's policy, given by

$$H(\pi^A(s^A_t)) = \frac{1}{2}(log(2\pi\sigma^2) + 1) \tag{10}$$

Both networks were trained using the RMSProp optimiser [35], using their respective losses.

To train the adversarial agent to find collisions against target policies it was trained using the adversarial reward function based on inverse headway given by:

$$r^A(s^A, a^A) = \min\left(\frac{1}{t_h}, 100\right) \tag{11}$$

where r^A is the adversary's reward, and the reward is capped at 100 to avoid the reward tending towards infinity as the headway approaches zero.

C. *Adversarially Robust Control (ARC)*

The Adversarially Robust Control framework utilises two networks, the Protagonist network P and the Adversary network A, initialised from the IL network (Section II-A) and ARL network (Section II-B), respectively. The scenario where both networks are learning to compete against each other can be formulated as a two player Markov Game, which is a multi-agent game theoretic formulation of an MDP [36], [37]. The Markov Game can be strictly competitive (zero-sum) or semi-competitive (nonzero-sum), depending on whether the agents are directly competing against each other or whether they have additional objectives [23]. The Markov Game with Protagonist P and Adversary A is denoted by a tuple ($\mathcal{S}$, $\mathcal{A}^A$, $\mathcal{A}^P$, $\mathcal{P}$, $\mathcal{R}^A$, γ). The P and A observe states $s^P_t \in \mathcal{S}$ and $s^A_t \in \mathcal{S}$ and take actions $a^P_t \in \mathcal{A}^P$ and $a^A_t \in \mathcal{A}^A$, respectively. The environment E then transitions to the next state according to transition model $\mathcal{P}$, and the adversary receives a reward $r^A_t \in \mathcal{R}^A$. Note, unlike RARL approaches with two RL agents, we do not define a reward for the Protagonist, rather the P network directly maximises the policy loss of the adversary, such that both agents are trained end-to-end using the same loss:

$$\begin{aligned} &\min_A \max_P \mathcal{L}_{\pi^A}(A, P) \\ &\quad = -log\pi^A(a^A_t | s^A_t) A(s^A_t, a^A_t) - \beta H(\pi^A(s^A_t)) \end{aligned} \tag{12}$$

Therefore, the aim of the Adversary is to maximise its reward function r^A, which encourages the agent to take actions which lead the following vehicle to collide into it. Meanwhile, the Protagonist aims to maximise this loss, effectively aiming to take actions which lead to lower rewards for the adversary, and thus less collisions. Having the P network directly maximise the Adversary's policy loss has the advantage that no additional training signal has to be engineered for the Protagonist (e.g. labels for supervised learning or rewards for reinforcement learning). This also makes the proposed framework more general, as it is agnostic to the learning technique used for pre-training (e.g. no assumptions about the stochasticity of the policy) and simply needs access to the weights of the P network. The Adversary used here differs from the one in Section II-B, in that it uses an additional observation, which is the action taken by the protagonist a^P. Therefore $s_t^A = (v, a, v_{rel}, t_h, a^P)$, making the output of the A network a function of the P network; $a^A = A(s^A) = A(P(s^P))$, and the policy loss $\mathcal{L}_{\pi^A}$ is differentiable with respect to both P and A. We train both networks in the highway driving scenario where the Protagonist controls the follower vehicle, whilst the Adversary controls the lead vehicle. Each training episode lasts for 5 minutes or until a collision occurs. The training is sped up by using the DNN-based simulator proxy described in [38], which acts as a type of World Model [39] estimating the simulator, and was shown to speed up training by up to a factor of 20. Further testing is later carried out in IPG CarMaker simulator to validate the control policy performance (Section III).

However, while naively maximising the policy loss in a strictly competitive game setting would lead to behaviours which degrade the performance of the adversary, it does not necessarily provide robust policies which generalise to different lead vehicle behaviours. We show that this type of competitive game setting causes the P agent to either learn an overly conservative driving strategy or to overfit to the adversarial lead vehicle while forgetting how to drive in non-adversarial scenarios. Instead, we propose to use a semi-competitive game setting where an auxiliary loss is used for training the P network, ensuring it does not overfit to the adversarial scenarios or catastrophically forget how to perform in its original state distribution.

The first possible issue with learning only from the adversary is becoming overly conservative to avoid collisions or overfitting to the adversarial scenarios created by the adversary. Since such driving scenarios represent edge-cases, which during normal driving would only occur rarely, there is the potential risk for the Protagonist to forget how to perform well in the natural driving scenarios. This is a similar issue to the catastrophic forgetting [40], [41], which can occur in domain adaption when the model adapts to a new domain and forgets the previous domain [42], [43]. Indeed, in Adversarial Policies, Gleave *et al.* [11] noted that fine-tuning target policies against adversaries leads RL policies to forget how to perform against normal opponents. Therefore, to avoid overfitting the P network to the adversarial scenarios, an auxiliary distillation loss $\mathcal{L}_D$ is defined which discourages the network from changing its behaviour drastically from the un-tuned IL model. This concept is similar to knowledge distillation [44] or policy distillation [45], however here the distillation loss is used to prevent catastrophic forgetting when training in a new distribution instead of distilling the policy into a smaller network. The $\mathcal{L}_D$ loss uses supervision from the un-tuned IL network by penalising the actions of the P model based on the absolute difference to the action which would have been taken by the original IL model for the same state:

$$\mathcal{L}_D = \|a^P - a^{IL}\| \tag{13}$$

Such that the final loss minimised by the Protagonist becomes:

$$\mathcal{L}_P = -\mathcal{L}_{\pi^A} + \lambda \mathcal{L}_D \tag{14}$$

where λ is a scaling hyperparameter.

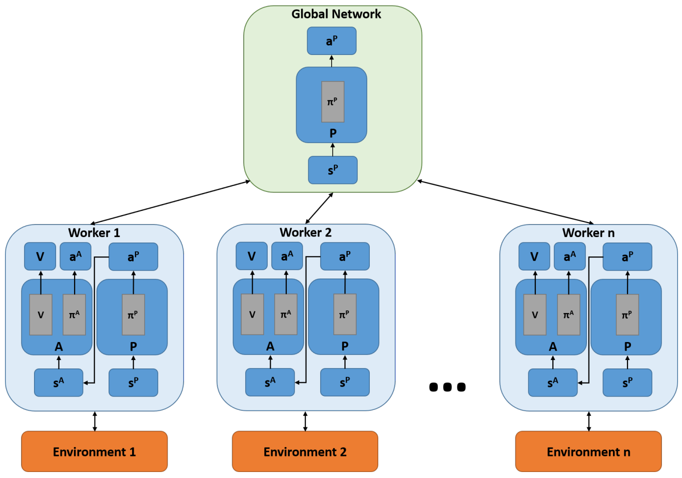

Fig. 1: Training environment.

A second possible overfitting issue with this framework is overfitting due to repetitive similar behaviour of the Adversary. Different from Adversarial Policies [11], which fine-tuned against fixed adversarial policies, we train both the P and A simultaneously, allowing the A to adapt as the P learns to counter it. However, this alone may not be enough, as the A network may get stuck in a local minima and continue to use the same strategy or it may adapt slowly to the improved robustness of the P network. Therefore, we train the P network in multiple environments simultaneously, where each environment E_i uses a different adversary A_i, where $i = \{1, 2, .., n\}$ for n total environments. The network updates calculated based on these environments are done asynchronously, using the Asynchronous Advantage Actor Critic (A3C) [32] formulation, where each instance of the simulation E_i copies the parameters of the global network to its own local network, where gradients are computed based on the experiences collected in E_i by the local network. The gradients are then used to update the global network, and the local network copies the new parameters from the global network. However, in our formulation the adversaries are different agents with different parameters θ^{A_i}, therefore the global network tracks the parameters of the P network, while each adversary A_i is updated in the local network only, as shown in Fig. 1. The Adversaries adapt to try to beat the Protagonist independently, allowing them to explore and learn different strategies, whilst the Protagonist is optimised against all Adversaries asynchronously.

TABLE I: ARC training parameters.

Parameter	Value
Adversary learning rate (actor), η_{actor}	1x10^{-4}
Adversary learning rate (critic), η_{critic}	1x10^{-2}
Protagonist learning rate, η_P	1x10^{-5}
Scaling parameter, λ	5x10^{4}
Discount factor, γ	0.99
Entropy coefficient, β	1x10^{-4}

III. Results

Using the described formulation, we pre-train 5 adversarial agents against the IL model for 2500 episodes. Then, we train the P and A_i networks end-to-end for 2500 episodes, experimenting with different number of environments $n = \{1, 5, 10, 25, 50\}$. The training hyperparameters can be found in Table I. As an ablation study, we also train 2 baselines to investigate the benefits of the suggested framework; ARC with a fixed single adversary and no $\mathcal{L}_D$ loss (ARC Adv. fixed, $\lambda = 0$) and ARC with a fixed single adversary (ARC Adv. fixed). We evaluate the performance during training, as well as the final trained control policies under two testing frameworks. Naturalistic driving tests the models in driving scenarios similar to those seen during training, and tests whether tuning the models against adversaries has degraded their performance in the original training distribution. The adversarial testing trains new adversaries against the control policy, and provides a measure of robustness against adversarial agents.

A. Training

The training results are shown in Fig. 2, where the mean step adversary rewards are visualised. Note, we show mean step reward instead of episode rewards/returns, as episodes with collisions can have significantly lower episode rewards as there are less steps to accumulate rewards. However, an episode with a collision is a successful episode for the adversary, and the higher mean step reward in such episodes reflects that. The rewards shown in Fig. 2 plot the performance during training with $n = 25$. It can be seen that the Adversary initially improves its performance against the Protagonist, with increasing step rewards in the first 1000 episodes. However, over the training process, the Protagonist becomes more robust, and the mean step rewards converge $r_t^A \approx 0.5$, which corresponds to a headway of 2s.

B. Validation

To understand the final performance of the fine-tuned policy, we employ two testing strategies for different driving conditions; *naturalistic driving* tests the control policy in typical driving conditions similar to those seen during imitation learning, and *adversarial testing* trains 5 new adversaries against the control policy and validates the robustness of the fine-tuned policy against adversarial agents and safety-critical edge case scenarios. The naturalistic testing is carried out in IPG CarMaker with different highway driving scenarios with lead vehicle velocities in the range [17, 40] m/s, acceleration [-6, 2] m/s^2, and road friction coefficient in [0.4, 1.0]. The adversarial testing trains 5 new agents against the target policy for 2500 episodes as described in Section II-B.

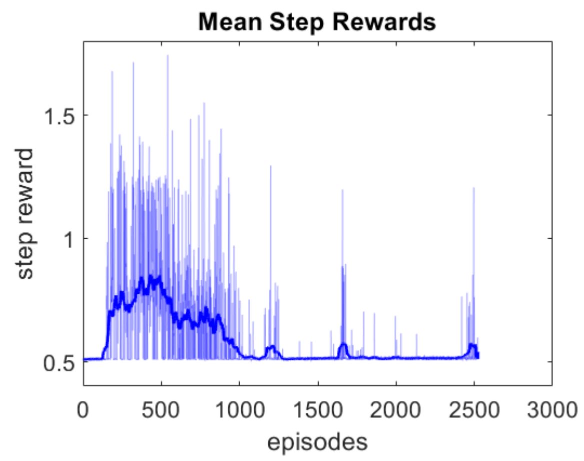

Fig. 2: Mean step rewards for the adversary during ARC training. The plot shows the running mean reward (with window size of 50), with the true rewards in the transparent plot.

The full results of both tests are shown in Table II. Firstly, we can see that the ARC with a fixed adversary and no distillation loss converges to an overly conservative driving behaviour, maintaining large distances from the vehicle in front, as shown by its average headway of 21s. Once the distillation loss is introduced, the ARC model with the fixed adversary is significantly less conservative with an average t_h of 3.3s. However, this model significantly overfits to the adversary it is training against, and fails to generalise to naturalistic driving as well as against new adversaries. Once the adversary is trained simultaneously with the protagonist, we see the model generalise to different scenarios significantly better. The ARC ($n = 1$) model can now drive without collisions with an average headway of 2.02s in naturalistic driving, as well as showing improved robustness against new adversaries when compared to the fixed adversary model. However, it is worth noting that the model still shows greater vulnerability to new adversaries compared to the original IL policy. Once we utilise multiple parallel environments ($n > 1$) with different adversaries, we obtain improved robustness to new adversaries compared to the IL policy, while also demonstrating similar level of performance in naturalistic driving. As illustrated in Fig. 3, the vulnerability of the ARC model to new adversaries reduces with increasing number n, up to 25. The minimum episode headway during the training of new adversaries for adversarial testing is illustrated in Fig. 4, which shows the significant improvement in robustness with ARC. While it would be expected that the robustness of ARC increases further with the size of n, our results show that the best robustness is reached at $n = 25$. A potential reason for the lower robustness with $n = 50$, is that the global number of episodes for each ARC model was fixed at 2500. This means that as the number of environments increases, each environment collects less experience in total,

TABLE II: Testing of final control policies under Natural (Nat.) and Adversarial (Adv.) Testing frameworks, with baseline comparison including Imitation Learning and different versions of Adversarially Robust Control.

Testing Framework	**Parameter**	**IL**	**ARC**	**ARC**	**ARC**	**ARC**	**ARC**	**ARC**	**ARC**
		[30]	**Adv. fixed, $\lambda = 0$**	**Adv. fixed**	**n = 1**	**n = 5**	**n = 10**	**n = 25**	**n = 50**
Nat. Testing	min. x_{rel} [m]	23.84	49.95	0.00	32.25	23.66	23.61	23.61	23.60
	mean x_{rel} [m]	57.37	584.76	81.81	59.78	57.35	57.35	57.35	57.36
	max. v_{rel} [m/s]	8.88	15.86	35.54	3.15	8.92	9.00	9.02	9.02
	mean v_{rel} [m/s]	0.0197	2.1350	0.0828	0.0368	0.0217	0.0205	0.0207	0.0211
	min. t_h [s]	1.74	1.97	0.00	1.55	1.74	1.74	1.74	1.74
	mean t_h [s]	1.99	21.08	3.30	2.02	1.99	1.99	1.99	1.99
	collisions	0	0	55	0	0	0	0	0
Adv. Testing	collisions against adversaries	800	0	2490	1150	456	224	78	320
	episodes until collision	245	-	3	16	538	532	1146	775

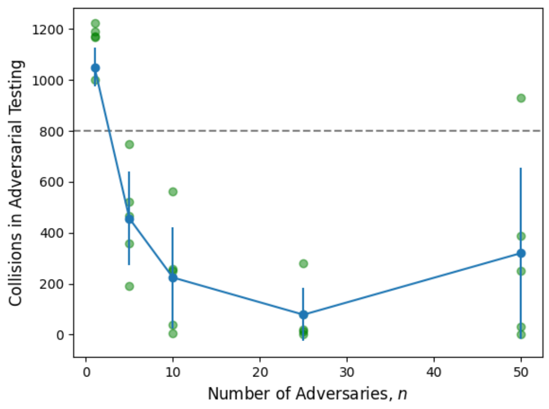

Fig. 3: Collisions for different number of adversaries during Adversarial Testing. Averaged over 5 training runs, individual collision numbers visualised by green markers, mean collisions by blue markers, and standard deviation by the error bars. The dashed line indicates the level of performance by the IL model before fine-tuning.

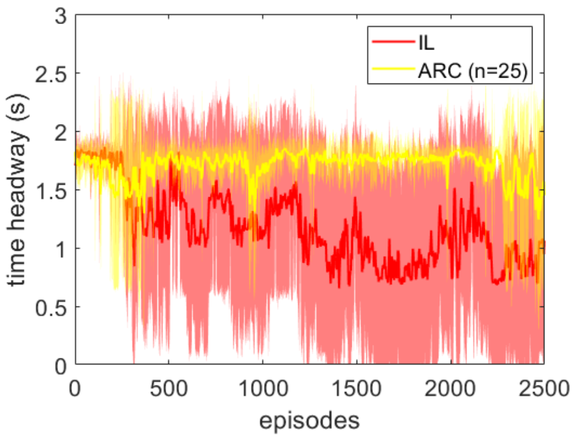

Fig. 4: Minimum episode headway during Adversarial Testing. Averaged over 5 training runs, with standard deviation shown in the shaded region.

and once the number of episodes per environment becomes too small there may not be enough experiences collected against the adversaries for the protagonist to learn how to counter them. This suggests there is a maximum number of environments that can be utilised for a given number of global training episodes. However, increasing the number of environments may still result in further improvement, if the number of global episodes is also increased.

The two testing frameworks demonstrate the benefit of the ARC approach. By starting with an initial policy susceptible to adversarial attack, and tuning it against adversarial policies, the policy becomes significantly more robust to such adversarial agents. Also, by utilising multiple environments in parallel, each using separate adversaries and training the policy asynchronously against all adversaries, the model gains superior generalisation and robustness. Furthermore, by utilising the distillation loss with knowledge from the IL network, the model avoids adapting overly conservative behaviour or overfitting to the adversarial scenarios, thereby ensuring the performance in the original training distribution is not degraded.

IV. Conclusions

In this paper, an approach to fine-tune the robustness and safety of a vehicle motion control policy was demonstrated. The approach was tested by fine-tuning an Imitation Learning control policy, which was shown to be vulnerable to adversarial agents. By training the IL policy against an ensemble of adversaries in multiple parallel simulations, it learned to counter the adversaries without overfitting to the behaviour of any single adversary. It was also demonstrated that after fine-tuning, the robustness to new adversaries is significantly improved, as demonstrated by the 90.25% reduction in collisions when tested against new adversarial agents. Moreover, testing in natural driving scenarios demonstrated that by utilising a distillation loss, the performance in the policy's original training distribution is not compromised. Therefore, this work demonstrated a fine-tuning strategy, which uses adversarial learning to significantly improve model generalisation and robustness to out-of-distribution scenarios, without trading off performance in its training distribution.

This work opens up multiple potential avenues for future work. Investigating this fine-tuning strategy for different control policies or use-cases would be interesting. Moreover, identifying techniques which could limit the amount of training with a simulator in the loop could be useful for reducing the training times and increasing the flexibility of this framework. This could be done by either improving the sample efficiency of the adversarial reinforcement learning used in the ARC framework, or extending the framework such that some or all of the training can be done offline with no simulator (e.g. by using a dataset of interactions between the adversary and protagonist). More importantly, further testing of the adversarially robust control in real-world training environments would be useful to gain further insight on how this framework could be expanded for real-world autonomous vehicles. This work has demonstrated that the technique is effective in improving the driving policies' robustness when leveraging multiple simultaneous parallel simulations. To extend this in the real-world, one option would be to leverage multiple pairs of physical protagonist and adversarial agents, which then update a global network. Alternatively, sim-to-real transfer, an active area of research [46]–[49], could be investigated to better leverage the faster training offered by simulators and minimising the amount of costly real-world training required.

Acknowledgment

This work was funded by the EPSRC under grant agreements (EP/R512217/1) and (EP/S016317/1).

References

[1] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[2] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3389–3396.

[3] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3357–3364.

[4] K. Katyal, K. Popek, C. Paxton, P. Burlina, and G. D. Hager, "Uncertainty-aware occupancy map prediction using generative networks for robot navigation," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5453–5459.

[5] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[6] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–9.

[7] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, "A survey of deep learning applications to autonomous vehicle control," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[8] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," in *2019 IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 9329–9338.

[9] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *2011 International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011, pp. 627–635.

[10] S. Kuutti, S. Fallah, and R. Bowden, "Training adversarial agents to exploit weaknesses in deep control policies," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 108–114.

[11] A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, and S. Russell, "Adversarial policies: Attacking deep reinforcement learning," *2020 International Conference on Learning Representations (ICLR)*, 2020.

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 2672–2680.

[13] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 4565–4573.

[14] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer, "Imitating driver behavior with generative adversarial networks," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 204–211.

[15] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[16] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[17] C. Xie, Y. Wu, L. v. d. Maaten, A. L. Yuille, and K. He, "Feature denoising for improving adversarial robustness," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 501–509.

[18] J. Morimoto and K. Doya, "Robust reinforcement learning," *Neural computation*, vol. 17, no. 2, pp. 335–359, 2005.

[19] A. Mandlekar, Y. Zhu, A. Garg, L. Fei-Fei, and S. Savarese, "Adversarially robust policy learning: Active construction of physically-plausible perturbations," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3932–3939.

[20] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary, "Robust deep reinforcement learning with adversarial attacks," *2018 International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2018.

[21] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," *arXiv preprint arXiv:1703.02702*, 2017.

[22] X. Pan, D. Seita, Y. Gao, and J. Canny, "Risk averse robust adversarial reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8522–8528.

[23] X. Ma, K. Driggs-Campbell, and M. J. Kochenderfer, "Improved robustness and safety for autonomous vehicle control with adversarial reinforcement learning," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1665–1671.

[24] M. Koren, S. Alsaif, R. Lee, and M. J. Kochenderfer, "Adaptive stress testing for autonomous vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1–7.

[25] V. Behzadan and A. Munir, "Adversarial reinforcement learning framework for benchmarking collision avoidance mechanisms in autonomous vehicles," *IEEE Intelligent Transportation Systems Magazine*, 2019.

[26] W. Ding, B. Chen, B. Li, K. J. Eun, and D. Zhao, "Multimodal safety-critical scenarios generation for decision-making algorithms evaluation," *arXiv preprint arXiv:2009.08311*, 2020.

[27] A. Corso, R. J. Moss, M. Koren, R. Lee, and M. J. Kochenderfer, "A survey of algorithms for black-box safety validation," *arXiv preprint arXiv:2005.02979*, 2020.

[28] S. Riedmaier, T. Ponn, D. Ludwig, B. Schick, and F. Diermeyer, "Survey on scenario-based safety assessment of automated vehicles," *IEEE Access*, vol. 8, pp. 87 456–87 477, 2020.

[29] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural computation*, vol. 3, no. 1, pp. 88–97, 1991.

[30] S. Kuutti, R. Bowden, H. Joshi, R. de Temple, and S. Fallah, "Safe deep neural network-driven autonomous vehicles using software safety cages," in *2019 20th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL)*. Springer, 2019, pp. 150–160.

[31] IPG Automotive GmbH, "Carmaker: Virtual testing of automobiles and light-duty vehicles," 2017. [Online]. Available: https://ipg-automotive.com/products-services/simulation-software/carmaker/

[32] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep

reinforcement learning," in *2016 International Conference on Machine Learning (ICML)*, 2016, pp. 1928–1937.

[33] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press Cambridge, 1998, vol. 135.

[34] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[35] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[36] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine learning proceedings 1994*. Elsevier, 1994, pp. 157–163.

[37] J. Perolat, B. Scherrer, B. Piot, and O. Pietquin, "Approximate dynamic programming for two-player zero-sum markov games," in *International Conference on Machine Learning*. PMLR, 2015, pp. 1321–1329.

[38] S. Kuutti, R. Bowden, H. Joshi, R. de Temple, and S. Fallah, "End-to-end reinforcement learning for autonomous longitudinal control using advantage actor critic with temporal context," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 2456–2462.

[39] D. Ha and J. Schmidhuber, "World models," *arXiv preprint arXiv:1803.10122*, 2018.

[40] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[41] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," *arXiv preprint arXiv:1312.6211*, 2013.

[42] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[43] H. Jung, J. Ju, M. Jung, and J. Kim, "Less-forgetful learning for domain expansion in deep neural networks," *arXiv preprint arXiv:1711.05959*, 2017.

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

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

[46] X. Pan, Y. You, Z. Wang, and C. Lu, "Virtual to real reinforcement learning for autonomous driving," *arXiv preprint arXiv:1704.03952*, 2017.

[47] A. A. Rusu, M. Večerík, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets," in *Conference on Robot Learning*. PMLR, 2017, pp. 262–270.

[48] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[49] B. Osiński, A. Jakubowski, P. Ziecina, P. Miłoś, C. Galias, S. Homoceanu, and H. Michalewski, "Simulation-based reinforcement learning for real-world autonomous driving," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 6411–6418.