Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 54-5 (2021) 223–228

Adversarial Learning of Robust and Safe Controllers for Cyber-Physical Systems

Luca Bortolussi [*,**] **Francesca Cairoli** [*]
Ginevra Carbone [*] **Francesco Franchina** [*] **Enrico Regolin** [*]

[*] *University of Trieste, Italy (e-mail: lbortolussi@units.it - francesca.cairoli@phd.units.it - ginevra.carbone@phd.units.it - francesco.franchina@studenti.units.it - eregolin@units.it).*
[**] *Saarland University, Saarbrücken, Germany.*

Abstract: We introduce a novel learning-based approach to synthesize safe and robust controllers for autonomous Cyber-Physical Systems and, at the same time, to generate challenging tests. This procedure combines formal methods for model verification with Generative Adversarial Networks. The method learns two Neural Networks: the first one aims at generating troubling scenarios for the controller, while the second one aims at enforcing the safety constraints. We test the proposed method on a variety of case studies.

Keywords: Robust control, Signal Temporal Logic, Adversarial Learning, Data-based Control, Test generation, Safe control.

1. INTRODUCTION

Controlling Cyber-Physical Systems (CPS) is a well-established problem in classic control theory [Howes et al., 2018]. State of the art solutions apply to all those models in which a complete knowledge of the system is available, i.e., scenarios in which the environment is supposed to follow deterministic rules. For such models a high level of predictability, along with good robustness, is achieved. However, as soon as these unpredictable scenarios come into play, traditional controllers are challenged and could fail. Ongoing research is trying to guarantee more flexibility and resilience in this context by using Deep Learning [Mnih, 2015] and, in particular, Reinforcement Learning for robust control. State of the art solutions perform reasonably well, but they still present evident limits in case of unexpected situations. The so called *open world scenarios* are difficult to model and to control, due to the significant amount of stochastic variables that are needed in their modelling and to the variety of uncertain scenarios that they present. Therefore, while trying to ensure safety and robustness, we need to be cautious about not trading them with model effectiveness.

In this work we investigate autonomous learning of safe and robust controllers in open world scenarios. Our approach consists in training two neural networks, inspired by Generative Adversarial Networks (GAN) [Goodfellow et al., 2014], that have opposite goals: the *attacker* network tries to generate troubling scenarios for the *defender*, which in turn tries to learn how to face them without violating some safety constraints. The outcome of this training procedure is twofold: on the one hand we get a robust controller, whereas on the other hand we get a generator of adverse tests. [1]

⋆ This work has been partially supported by the PRIN project "SEDUCE" n. 2017TWRCNB.

[1] Code is available at: `https://github.com/ginevracoal/adversarialGAN/`

The learned controller is a black-box device capable of dealing with adverse or unobserved scenarios, though without any worst-case guarantee. In this regard, one could complement our method with a shield-based approach, as proposed e.g. by Avni et al. [2019].

2. PROBLEM STATEMENT AND RELATED WORK

Safety of a system can be formalised as the satisfaction of a set of safety requirements. A popular approach to mathematically express such safety requirements is *Signal Temporal Logic* (STL) [Donzé and Maler, 2010]. Temporal logic is used in the context of *formal verification* to express the desired behaviour of a system in terms of time. It extends propositional logic with a set of modal operators capturing the temporal relation among events [Goranko and Rumberg, 2020]. STL formulas, in particular, deal with properties of continuous-time real-valued signals, such as CPS trajectories. In our application, we express safety requirements only by means of time-bounded formulas over fixed-length trajectories. In particular, we rely on STL *quantitative semantics*, which is capable of capturing, for each trajectory, the level of satisfaction of the desired property by measuring how much the input trajectory can be shifted without changing its truth value. Such measure is often referred to as *robustness* and it is exploited in this work as the objective function of an optimization problem.

We model the interaction of an agent with an adversarial environment as a *zero-sum game*, similarly to the strategy behind GANs. The concept of zero-sum game is borrowed from game theory and denotes those situations in which one player's gain is equivalent to another's loss. In such situations, the best strategy for each player is to minimize its loss, while assuming that the opponent is playing at its best. This concept is known in literature as *minmax strategy*. In practice, we use GAN architectural and theoretical design to reach two main objectives: a controller, that safely acts under adverse conditions, and an attacker, which gains insights about troubling

10.1016/j.ifacol.2021.08.502

scenarios for the opponent. The concept is closely related to that of *Robust Adversarial Reinforcement Learning (RARL)* [Pinto et al., 2017], a Reinforcement Learning framework, involving an agent and a destabilizing opponent, that is robust to adverse environmental disturbances. In this case the term "robustness" does not refer to Signal Temporal Logic, but instead, to the cumulative reward computed on the learned policy with respect to the varying test conditions. Other recent RL techniques involving STL constraints include Balakrishnan and Deshmukh [2019], Bozkurt et al. [2020], Liu et al. [2021]

3. METHODOLOGY

Agent-Environment Model. Due to coexistence of continuous and discrete components, CPSs are typically represented as hybrid models: the continuous part is represented by differential equations that describe the behaviour of the plant; the discrete part, instead, identifies the possible states of the controller. We decompose our model in two interacting parts: the *agent* a and the *environment* e. Both of them are able to observe at least part of the whole state space $\mathcal{S}$, i.e. they are aware of some observable states $\mathcal{O}\subset\mathcal{S}$. By distinguishing between the observable states of the agent $\mathcal{O}_a\subseteq\mathcal{O}$ and of the environment $\mathcal{O}_e\subseteq\mathcal{O}$, we are able to force uneven levels of knowledge between them. Notice that the observable states, for both the agent and the environment, could also include environmental variables involved in the evolution of the system.

Let $\mathcal{U}_a$ and $\mathcal{U}_e$ be the spaces of all possible actions for the two components. We discretize the evolution of the system as a discrete-time system with step Δt, which evolves according to a function $\psi:\mathcal{S}\times\mathcal{U}_a\times\mathcal{U}_e\times\mathbb{R}\longrightarrow\mathcal{S}$. By taking control actions at fixed time intervals of length Δt, we obtain a discrete evolution of the form $s_{i+1}=s_i+\psi(s_i,u_a^i,u_e^i,t_i)$, where $t_i:=t_0+i\cdot\Delta t$, $u^i:=u(t_i)$ and $s_i:=s(t_i)$. Therefore, we are able to simulate the entire evolution of the system over a time horizon H via ψ and to obtain a discrete trajectory $\xi=s_0...s_{H-1}$.

STL syntax. The STL syntax is defined by

$$\varphi:=\texttt{true}\,|\,f(s)>0\,|\,\neg\varphi\,|\,\varphi_1\wedge\varphi_2\,|\,\varphi_1\mathtt{U}_{[a,b]}\varphi_2$$

where $s:\mathbb{R}^+\to\mathcal{S}$ is a signal, $f:\mathcal{S}\to\mathbb{R}$ is a real-valued function, and $[a,b]$ is an interval of non-negative real numbers in the time domain $\mathbb{R}^+$. Two important temporal operator can be derived from the syntax above: the *eventually* operator $\Diamond_{[a,b]}\varphi\equiv\texttt{true}\mathtt{U}_{[a,b]}\varphi$, and the *globally* operator $\Box_{[a,b]}\varphi\equiv\neg(\Diamond_{[a,b]}\neg\varphi)$. The definition of Boolean and quantitative semantics is given in Section C of the Appendix.

A safety requirement is expressed as an STL formula φ; we call h its temporal depth. [2] The robustness of a trajectory quantifies the level of satisfaction w.r.t. φ and it determines how safe the system is in that configuration. Robustness is denoted as a function $R_\varphi:\mathcal{S}^h\to\mathbb{R}$, measuring the maximum perturbation that can be applied to a given trajectory of length h without changing its truth value w.r.t. φ. It is straightforward to use this measure as the objective function in our minmax game.

Multi-objective formulation. In case of multiple safety requirements, we define a different STL formula for each of these requirements and likewise we compute the respective robustness values. As a matter of fact, the order of magnitude of each robustness value depends on the order of magnitude of the CPS variables involved. Therefore, a single STL formula that combines the safety requirements all together may result in an unbalanced objective function, skewed towards some components that are not necessarily the most safety-critical. To overcome this problem we normalize the variables involved and we define the objective function as a weighted sum of the robustness values R_{φ_i} resulting from each requirement φ_i. Let $\Phi=\{\varphi_1,...,\varphi_m\}$ denote a set of m safety requirements, the combined robustness score R_Φ is defined as:

$$R_\Phi(\cdot):=\frac{1}{\alpha}\sum_{i=1}^{m}\alpha_i\cdot R_{\varphi_i}(\cdot), \tag{1}$$

where $\alpha=\sum_{i=1}^m\alpha_i$. By tuning the weights $\alpha_1,...,\alpha_m$ we are able to explicitly influence the importance of each factor in the training objective. The choice of these hyper-parameters will be case-specific. This formulation is typically used in multi-objective optimization scenarios [Li et al., 2016]. In (1), we assume w.l.o.g. for notational simplicity that each formula φ_i has the same time depth h. However, our framework can be straightforwardly extended to more general STL properties with different time depths. *Attacker-Defender architecture.* The proposed framework builds on GAN architectural design, in which two NNs compete in a minmax game to reach opposite goals. One network, denoted by A, represents the *attacker*, while the other, denoted by D, represents the *defender*. The aim of the former is to generate environment configurations in which the defender is not able to act safely, whereas, the latter tries to keep the CPS as safe as possible. In practice, the defender D can be interpreted as a controller for the agent.

Optimization strategy. Given a time horizon H, an initial state s_0, meaning the state at time t_0, and two sequences of actions $\mathbf{u}_a=(u_a^0,...,u_a^{H-1})$ and $\mathbf{u}_e=(u_e^0,...,u_e^{H-1})$, one for the agent and one for the environment, it follows that the evolution of a trajectory ξ is obtained by evaluating ψ at each time steps $t_i\in\{t_0,...,t_{H-1}\}$. The minmax problem can be expressed as finding the sequences $\mathbf{u}_e$ and $\mathbf{u}_a$ that solve

$$\min_{\mathbf{u}_e}\max_{\mathbf{u}_a}[\mathcal{J}(s_0,\mathbf{u}_a,\mathbf{u}_e)]. \tag{2}$$

The objective function $\mathcal{J}$ is the cumulative sum of the robustness scores computed at each timestep during the generation of the whole trajectory ξ on the sub-trajectory $\xi[t,t+h-1]$ available at timestep t, i.e.

$$\mathcal{J}(s_0,\mathbf{u}_a,\mathbf{u}_e):=\sum_{t=0}^{H-h}R_\Phi(\xi[t,t+h-1]).$$

In our setting, the sequences of actions are iteratively determined by the two adversarial networks. In particular, let θ_A be the weights of the attacker's network A and θ_D the weights of the defender's network D. At each timestep, the attacking network,

$$\begin{aligned}A:\Theta_A\times\mathcal{O}_e\times\mathcal{Z}&\longrightarrow\mathcal{U}_e\\(\theta_A,\mathbf{o}_e,\mathbf{z})&\longmapsto u_e,\end{aligned}$$

receives the current observable state of the environment $\mathbf{o}_e$, the noise coefficient $\mathbf{z}$ and outputs the coefficients u_e, defining the adversarial environmental components. Similarly, the defender network,

$$\begin{aligned}D:\Theta_D\times\mathcal{O}_a&\longrightarrow\mathcal{U}_a\\(\theta_D,\mathbf{o}_a)&\longmapsto u_a,\end{aligned}$$

reads the current observable state of the agent $\mathbf{o}_a$ and produces the control action u_a.

[2] The temporal depth of a formula is defined recursively as the sum of maximum bounds of nested temporal operators.

To ease the notation, we introduce a function

$$\psi_t : \mathcal{S} \times \Theta_D \times \Theta_A \longrightarrow \mathcal{S}^t$$
$$(s_0, \theta_D, \theta_A) \longmapsto s_0 ... s_{t-1},$$

which iteratively applies ψ for each pair of actions

$$u_a^j = D(\theta_D, \mathbf{o}_a^j)$$
$$u_e^j = A(\theta_A, \mathbf{o}_e^j, \mathbf{z}),$$

where $j \in \{0,...,t-1\}$ indexes the simulation interval.

The formalism introduced by the two policy networks transforms the problem of finding the best sequences of actions, $\mathbf{u}_a$ and $\mathbf{u}_e$, to that of finding the best networks' parameters, θ_D and θ_A. This leads to the objective

$$J(s_0, \theta_A, \theta_D) = \sum_{t=0}^{H-h} R_\Phi[\psi_h(s_t, \theta_D, \theta_A)] \tag{3}$$

and the minmax game $\min_{\theta_A} \max_{\theta_D} J(s_0, \theta_A, \theta_D)$ is now directly expressed in terms of the training parameters.

In such setting, the defender aims at generating safe actions by tuning its weights in favour of a maximization of the objective function, i.e., a maximization of the cumulative robustness score. The attacker, on the other hand, aims at generating troubling scenarios for the opponent by minimizing the objective function, i.e., minimizing the cumulative robustness score.

The horizon H represents the number of simulation steps performed while keeping the parameters θ_A and θ_D fixed. In principle, we could choose $H = h$, without the need of having a summation in (3), possibly taking a larger time bound h in the formulae of Φ. However, this would make the objective excessively rigid. In fact, if a controller would work well everywhere but on a small sub-region of the trajectory, such an objective would return a penalization also for the regions where the controller performs well. This effect is avoided by considering $h \ll H$ in the objective (3).

Testing phase. In the testing phase, we generate a trajectory of length H and we check separately each safety requirement on such trajectory, in particular we check that the requirement φ_i is globally satisfied, i.e., the condition $\Box_{[0,H]}\varphi_i$. Therefore, for each property, a positive value of robustness at test time means that the requirement is met during the whole evolution of the system w.r.t. the time horizon H. The training and testing pseudo-codes are shown in Section A of the Appendix.

4. EXPERIMENTS

We test the proposed architecture on two different case studies: a cart-pole balancing problem and a platooning problem. Both systems are embedded into environments with a stochastic evolution.

The Attacker-Defender networks are trained against each other for a given number of epochs. Once the training is over, the performances of the trained Defender are tested in two different ways. On one hand, we generate a test set containing 1k different initial configurations, uniformly sampled from predefined compact sets (see Tab. B.1 and B.2 in the Appendix). From each of these points we generate a trajectory evolving according to the trained Attacker and Defender networks, then check each requirement separately on each trajectory as specified in the previous section. The second approach to evaluate the performances of the trained Attacker-Defender network is to consider an environment that evolves unaware of the state of the system. In both cases, we compare the performance of Defender with that of a classical controller.

Hyperparameter tuning is necessary for the GAN architecture to achieve the desired performances in terms of safety. The choice of the architecture (number and size of the layers), the training hyperparameters, the time horizon H and weights for the cumulative robustness have a strong impact on the final results. In particular, the number of training iterations performed by the Attacker network and by the Defender network has a strong impact on the performances of the trained networks. By tuning this number we are able to ensure that the Attacker is strong enough to generate challenging configurations of the environment, without preventing the Defender network from learning a secure controller. We performed manual tuning on a combination of hyperparameters and architectures, however one could also automate this process by maximizing the percentage of safe trajectories produced by the learned controller.

4.1 Cart-Pole balancing

The *Cart-Pole* system [Florian, 2005] (also known as Inverted Pendulum) consists of a cart and a vertical pole attached to the cart by an unactuated joint. The cart is allowed to move along the horizontal axis, while the pole moves is the vertical plane parallel to the track. The goal is to keep the pole balanced by learning an optimal policy for the cart, which influences the swinging movement of the pole. This problem is a well known benchmark in both classical control [Aguilar-Ibáñez et al., 2014, Liu et al., 2008] and reinforcement learning [Nagendra et al., 2017, Lillicrap et al., 2019] applications.

Moving target and track-cart friction. In order to test the full potential of our framework, we consider a complex stochastic environment made of a moving target for the cart to follow and a friction coefficient between the cart and its track. These two components, governed by the Attacker network, represent the two potentially adversarial components of the system.

Model. The observable states $\mathbf{o}_a$ for the Defender and $\mathbf{o}_e$ for the Attacker are: cart position x, cart velocity $\dot{x}$, pole angle θ, pole angular velocity $\dot{\theta}$ and target position $\hat{x}$. Given $\mathbf{o}_e$, the Attacker's policy network A generates the adverse coefficients, i.e., friction μ and target position $\hat{x}$, both constrained to assume realistic values w.r.t. the physical settings of our application. The Defender reads the current state $\mathbf{o}_a$ and generates the desired control action f for the cart, which is meant to keep the pole balanced during the whole trajectory. The dynamic of the system is described by the following equations [Wang, 2011]:

$$\begin{cases} \ddot{x} &= \dfrac{f - \mu\dot{x} + m_p l \dot{\theta}^2 \sin\theta - m_p g \cos\theta \sin\theta}{m_c + m_p \sin\theta^2} \\ \ddot{\theta} &= \dfrac{g\sin\theta - \cos\theta \ddot{x}}{l} \end{cases}, \tag{4}$$

where m_p is the mass of the pole, m_c is the mass of the cart, l is half the pole length and g is the gravitational constant.

We impose the STL requirement $\varphi_d = \Box_{[0,h]}(d \leq d_{\max} \wedge d \geq d_{\min})$ on the distance $d = \|x - \hat{x}\|$ between the cart and its target, where $d_{\min}$ and $d_{\max}$ are the minimum and maximum distances allowed. Similarly, we set the requirement $\varphi_\theta = \Box_{[0,h]}(\theta \leq \theta_{\max} \wedge \theta \geq \theta_{\min})$ on the angle.

The objective function is the combination of two cumulative robustness components, one on the distance, R_{φ_d}, and one

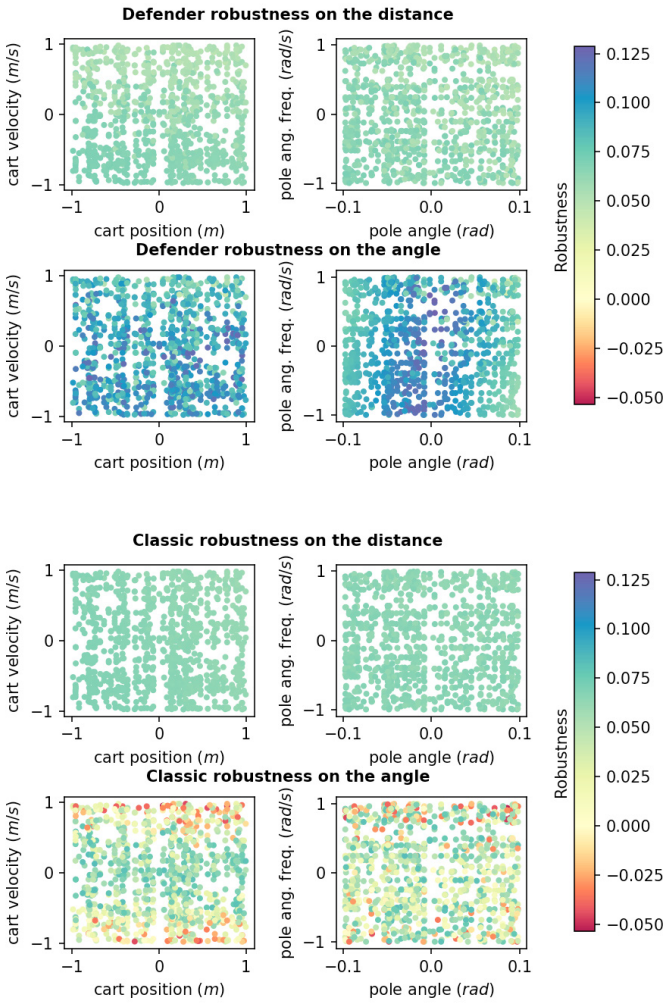

Fig. 1. Cartpole balancing problem in which the friction and the moving target are generated by the Attacker network. This plot shows the robustness values achieved by the Defender and the classic controller. Robustness differences are computed separately for the two requirements imposed on the distance and on the angle. Trajectories start from 1k random initial states and evolve on a time horizon $H = 200$ with a step of $\Delta t = 0.05s$, resulting in $\Delta t \cdot H = 10\,s$ long simulations.

on the angle, R_{φ_θ}, whose contributions are weighted by a coefficient $\alpha \in [0,1]$:

$$J(s_0,\theta_A,\theta_D) = \alpha \sum_{t=0}^{H-h} R_{\varphi_d}[\xi_t] + (1-\alpha) \sum_{t=0}^{H-h} R_{\varphi_\theta}[\xi_t],$$

where $\xi_t = \psi_h(s_t,\theta_D,\theta_A)$.

Results. The experimental settings are presented in Table B.1 of the Appendix. Fig. 1 shows that the trajectories evolving according to the Defender network all achieved positive robustness, despite the adversarial reactive environment governed by the Attacker. Moreover, Fig. 2 shows the evolution of the system in a fixed environmental setting, for two different controllers: the Defender network and a classical robust controller based on a Sliding Mode Control (SMC) architecture [Edwards and Spurgeon, 1998]. The Defender is

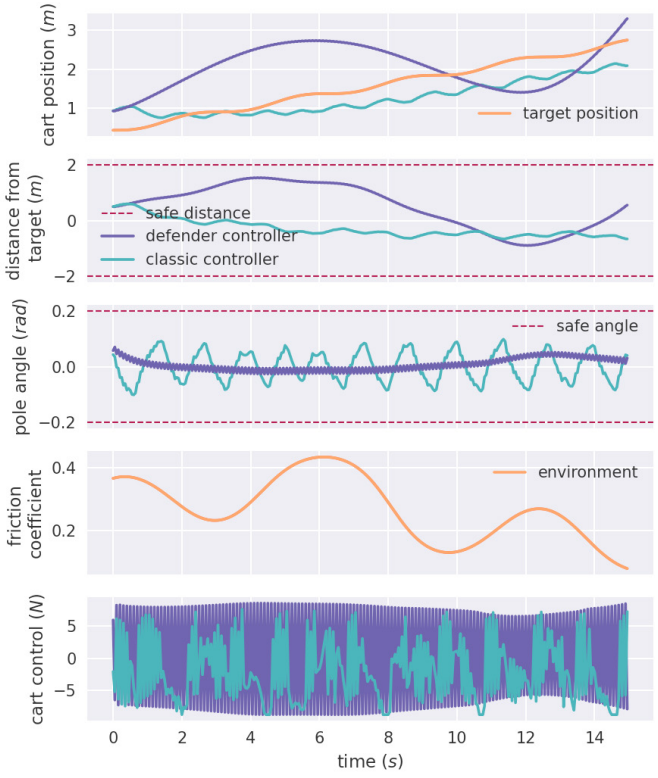

Fig. 2. Sample of the system evolution for the cartpole balancing problem, using a classical controller and the Defender network against the same environment, represented by fixed trajectories of target positions and the cart-track friction coefficients. The *defender* cart is controlled by the Defender network, while the *classic* one is managed a classical controller. The common initial state is: cart position $x = 0.9055\,m$, cart velocity $\dot{x} = 0.0348\,m/s$, pole angle $\theta = 0.0808$ *rad*, pole angular velocity $\dot{\theta} = 0.0399$ *rad*/*s*. The trajectory evolves on a time horizon $H = 300$ with time step size $\Delta t = 0.05\,s$ ($\Delta t \cdot H = 15\,s$ long simulation).

able to maintain safety during the whole trajectory on both θ and d, adequately counteracting the cart-track friction, while the classical controller has a worse overall performance and in a few cases failed to guarantee safety within the specified initialization grid. In this setting the Defender exhibits chattering in its control signal. In fast evolving systems, such as cartpole, this phenomenon could be avoided by including a regularization term on the control signal during the training phase. It should be noticed that the relatively low sampling frequency of $20Hz$ could also affect the SMC based controller.

4.2 Car platooning

A *platoon* [Banjanovic-Mehmedovic et al., 2018] is a group of vehicles travelling together very closely and safely. This problem is usually faced with techniques that coordinate the actions of the entire pool of vehicles as a single entity [Jia et al., 2016]. This approach, though, requires specific hardware and a distributed system of coordination that might be difficult to realise in complex scenarios. Our method, instead, builds a robust controller for individual decision-making, hence it fits into the autonomous driving field. In this setting, we assume that all vehicles are equipped with an hardware component

called *LIDAR scanner*, which is able to measure the distance between two cars by using a laser beam. In the basic scenario, only involving two cars, the car in front is called the *leader* and acts according to the Attacker network, while the second one is the *follower*, whose behaviour is determined by the Defender network. This setting trivially extends to the case of n cars, where the first car is the leader and the other ones all act as followers, controlled by the same Defender.

Platooning with Power Consumption

An additional problem that can be addressed in the platooning problem is the optimization of the energy consumption of the follower car, similarly to what has been proposed by [Zambelli and Ferrara, 2019], who exploited a non-cooperative distributed MPC framework.

We are given a vehicle with mass m and effective wheel radius R_e, which moves on a flat straight line covering the distance $x(t)$, with the driver inputs producing a torque at wheel level T_w. We factor the loss in two terms, which include the effects of rolling resistance (deformation of the rolling wheel on the ground) and aerodynamic resistance. The dynamics follows the following equation

$$m\ddot{x}(t) = \frac{T_w(t)}{R_e} - C_r mg\dot{x}(t) - \frac{1}{2}\rho C_a S\dot{x}^2(t), \tag{5}$$

where C_r is the rolling resistance coefficient, g the gravitational acceleration, ρ the air density, C_a the aerodynamic coefficient and S the equivalent vehicle surface.

The torque at wheel level $T_w(t)$ is a function of time given by the combined effect of electrical torque at motor level, T_m, and the one due to the conventional brake action, T_b, that acts directly on the brake calipers. By factoring in the gear ratio between motor and wheels, one has:

$$T_w(t) = T_m(t)\cdot r_g + T_b(t). \tag{6}$$

From an energy efficiency standpoint, it is reasonable to expect an optimal platooning policy to minimize the overall consumed electrical energy. This can be achieved by operating as much as possible the electric powertrain in its most efficient working point, and by avoiding situations in which the conventional brake has to be operated for safety reasons, i.e. when the safety requirement on the distance is violated.

With regards to powertrain efficiency, we define an efficiency map $\eta(T_m(t),\omega_m(t))$ which combines the effects of battery and e-motor, where ω_m is the motor speed. At time t the consumed electric power is

$$P_m(t) = T_m(t)\omega_m(t)\eta(T_m(t),\omega_m(t))^{-sign(T_m(t))}. \tag{7}$$

In a single-gear setting, the motor speed is related to vehicle speed through the relation $\omega_m(t) = \frac{\tau_g}{R_e}\dot{x}(t)$.

Model. We consider the simple case of two cars, one *leader* l and one *follower* f, whose internal states are position x, velocity v and acceleration a. The follower f acts as the agent of this system, while the leader l is considered to be part of the adversarial environment, to simulate a cyberattack scenario. They have the same observable states $\mathbf{o}_a = \mathbf{o}_e = (\dot{x}_l, \dot{x}_f, d)$, given by their velocities and by their relative distance d. In the basic platooning setting the policy networks A and D output the accelerations $\mathbf{u}_a = \ddot{x}_f$ and $\mathbf{u}_e = \ddot{x}_l$, which are used to update the internal states of both cars. When the energy consumption evaluation is involved, the policy networks

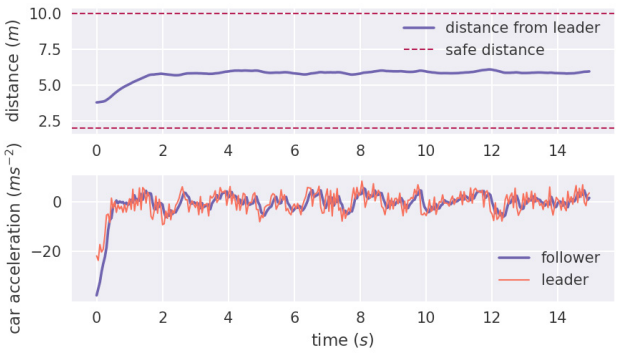

Fig. 3. Defender network (follower) against Attacker network (leader) for car platooning problem. Initial state: distance between cars $d = 4.2439\ m$, leader velocity $v_l = 18.1229 m/s$, follower velocity $v_f = 15.7211 m/s$. The time horizon is $H = 300$ and the step width is $\Delta t = 0.05s$.

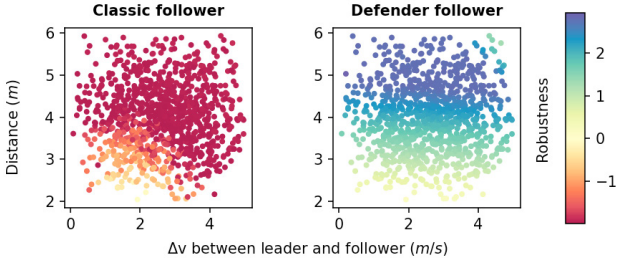

Fig. 4. Global robustness values on the distance requirement for a classic follower and the Defender follower against the Attacker leader, starting from $1k$ different random configurations of the system (leader and follower cars positions and velocities). Random initializations of the states are described in Tab. B.2 of the appendix. The time horizon is $H = 200$ and the step width is $\Delta t = 0.05s$. c

output electric and conventional torque values for the two cars, $\mathbf{u}_a = (T^a_{el}, T^a_b), \mathbf{u}_e = (T^e_{el}, T^e_b)$, that are used to compute the corresponding accelerations. The dynamic of a car with mass m and velocity v is described as $m\frac{dv}{dt} = ma_{\text{in}} - \nu mg$, where a_{in} is the input acceleration produced by one of the two policies, ν is the friction coefficient and g is the gravity constant. We impose the following STL requirements: $\varphi_d = \square_{[0,h]}(d \leq d_{\max} \wedge d \geq d_{\min})$ on the distance $d = x_l - x_f$ between the two vehicles and $\varphi_e = \square_{[0,h]}(e \leq e_{\max})$ on the power energy consumption e (note that the higher the robustness of φ_e, the lower the energy consumption). The objective of our optimization problem is

$$J(s_0,\theta_A,\theta_D) = \alpha \sum_{t=k}^{H-1} R_{\varphi_d}[\xi_t] + (1-\alpha)\sum_{t=k}^{H-1} R_{\varphi_e}[\xi_t],$$

where $\xi_t = F_h(s_t,\theta_D,\theta_A)$, R_{φ_d} is the robustness on the distance d and R_{φ_e} is the robustness of the energy consumption. The extension to a platoon of n cars is straightforward: the first car is the leader and each of the other cars follows the one in front. The first pair of subsequent cars acts as described in the two-cars model, while the other followers are controlled by copies of the same Defender network.

Results. In the basic platooning scenario, i.e. ignoring energy consumption, the leader acts according to the Attacker's policy,

with sudden accelerations and brakes. In such case, the follower learns to manage the unpredictable behaviour of the attacker by maintaining their relative distances within the safety range, as shown in Fig. 3.

Introducing requirements on the energy consumption makes the problem of platooning more challenging. Nonetheless, our architecture is still able to provide a safe controller. Fig. D.1 in Sec. D of the Appendix shows an evolution of the system in which the leader car is unaware of the state of the follower and the two followers are managed by the Defender network and by a PID classical controller, which implements a basic "energy aware" distance-tracking strategy. In this scenario, both controllers are able to ensure safety during the whole trajectory, although at the moment the defender appears to be focusing mostly on keeping the ideal distance from the leader, even applying an energy-inefficient strategy.

However, when the leader actions are adverse reactions to the state of the follower, meaning actions chosen by the Attacker, the global robustness achieved by the Defender is in general much higher than the one achieved by the classic controller, as shown in Fig.4.

5. CONCLUSIONS

Classical control theory struggles in giving adequate safety guarantees in many complex real world scenarios. New reinforcement learning techniques aim at modelling the behaviour of complex systems and learning optimal controllers from the observed data. Therefore, they are particularly suitable for stochastic optimal control problems where the transition dynamics and the reward functions are unknown. We proposed a new learning technique, whose architecture is inspired by Generative Adversarial Networks, and tested its full potential against the vehicle platooning and the cart-pole problem with additional stochastic components. We tested the learned controllers against black-box adversarial policies in the case of completely observable systems, but our framework could also be straightforwardly extended to partially observable systems. Our approach has been able to enforce safety of the model, while also gaining insights about adverse configurations of the environment. As future work, we plan to improve the performances in the multi-objective case, to test more complex scenarios, to investigate the scalability of this approach, and to introduce a regularization in the objective function in order to make the Defender policy less stiff, removing chattering behaviour.

REFERENCES

Aguilar-Ibáñez, C., Mendoza-Mendoza, J., and Dávila, J. (2014). Stabilization of the cart pole system: by sliding mode control. *Nonlinear Dynamics*, 78(4), 2769–2777.

Avni, G., Bloem, R., Chatterjee, K., Henzinger, T.A., Könighofer, B., and Pranger, S. (2019). Run-time optimization for learned controllers through quantitative games. In *CAV 2019*, 630–649. Springer.

Balakrishnan, A. and Deshmukh, J.V. (2019). Structured reward shaping using signal temporal logic specifications. In *2019 IEEE/RSJ IROS*, 3481–3486. IEEE.

Banjanovic-Mehmedovic, L., Butigan, I., Mehmedovic, F., and Kantardzic, M. (2018). Hybrid automaton based vehicle platoon modelling and cooperation behaviour profile prediction. *Tehnicki vjesnik - Technical Gazette*, 25(3).

Bozkurt, A.K., Wang, Y., Zavlanos, M., and Pajic, M. (2020). Model-free reinforcement learning for stochastic games with linear temporal logic objectives. *arXiv preprint arXiv:2010.01050*.

Donzé, A. and Maler, O. (2010). Robust satisfaction of temporal logic over real-valued signals. In K. Chatterjee and T.A. Henzinger (eds.), *FORMATS*. Springer Berlin Heidelberg.

Edwards, C. and Spurgeon, S. (1998). *Sliding mode control: theory and applications*. Crc Press.

Florian, R. (2005). Correct equations for the dynamics of the cart-pole system.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *NeurIPS*, 2672–2680.

Goranko, V. and Rumberg, A. (2020). Temporal logic. In E.N. Zalta (ed.), *SEP*. Stanford University, spring 2020 edition.

Howes, S., Mohler, I., and Bolf, N. (2018). Multivariable identification and pid/apc optimization for real plant application. In *ACHEMA*.

Jia, D., Lu, K., Wang, J., Zhang, X., and Shen, X. (2016). A survey on platoon-based vehicular cyber-physical systems. *IEEE Communications Surveys & Tutorials*, 18(1), 263–284.

Li, X., Vasile, C.I., and Belta, C. (2016). Reinforcement learning with temporal logic rewards. *CoRR*, abs/1612.03471.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2019). Continuous control with deep reinforcement learning.

Liu, W., Mehdipour, N., and Belta, C. (2021). Recurrent neural network controllers for signal temporal logic specifications subject to safety constraints. *IEEE Control Systems Letters*.

Liu, Y., Yu, H., Wane, S., and Yang, T. (2008). On tracking control of a pendulum-driven cart-pole underactuated system. *IJMIC*, 4(4), 357–372.

Mnih, V.e.a. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533.

Nagendra, S., Podila, N., Ugarakhod, R., and George, K. (2017). Comparison of reinforcement learning algorithms applied to the cart-pole problem. In *ICACCI 2017*, 26–32. IEEE.

Pinto, L., Davidson, J., Sukthankar, R., and Gupta, A. (2017). Robust adversarial reinforcement learning. In *ICML*, 2817–2826. PMLR.

Wang, J.J. (2011). Simulation studies of inverted pendulum based on pid controllers. *Simulation Modelling Practice and Theory*, 19, 440–449.

Zambelli, M. and Ferrara, A. (2019). Robustified distributed model predictive control for coherence and energy efficiency-aware platooning. In *ACC 2019*. IEEE.