2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W) | 978-1-6654-3950-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/DSN-W52860.2021.00038

RADICS: Runtime Assurance of Distributed Intelligent Control Systems

Brian Wheatman
Johns Hopkins University
wheatman@cs.jhu.edu

Jerry Chen
Johns Hopkins University
gchen41@jhu.edu

Tamim Sookoor
Johns Hopkins Applied Physics Lab
tamim.sookoor@jhuapl.edu

Yair Amir
Johns Hopkins University
yairamir@cs.jhu.edu

***Abstract*—We describe RADICS: Runtime Assurance of Distributed Intelligent Control Systems, which combines a Simplex-based, black-box monitor with a white-box monitor to ensure correct behavior and good performance of AI systems. The black-box monitor allows the system to detect when the AI controller is on a failing trajectory and use a provably safe, but less performant algorithm, to right the system. The white-box monitor predicts when the AI controller will be put on such a trajectory before it happens and helps maximize the performance of the overall system. We describe the overall approach in detail and implement a simple version of it on a case study into controlling the lights in a small traffic grid.**

***Index Terms*—Assured AI, Reinforcement Learning Dependability, Traffic Light Controller**

I. Introduction

The majority of the world population is projected to live in urban areas by 2050. To support this rapid increase, we need to modernize our aging infrastructure with controllers that can constantly optimize the system's performance, such as with machine learning techniques. In recent years, AI/ML deep neural networks (DNN) have brought dramatic improvements to diverse tasks such as automatic speech recognition, natural language processing, image recognition, medical image analysis, bioinformatics, and autonomous driving. One of the main limitations of these techniques is their opaque failure modes: it is difficult to understand exactly how these systems work and predict when and how they will fail.

In the vast majority of cases, these systems yield incredible results, much better than what was possible just a few years ago. However, in rare cases, they fail spectacularly in unexpected ways, often in ways that are hard for humans to accept. It looks like the AI system that worked perfectly in so many complex situations failed miserably in a case that looks obvious to the human eye [1]. Even if that error can be fixed in the DNN through additional learning, it is not clear how to generalize this concept to other potential errors, and the suspicion is that the distribution of these erroneous edge cases is such that no amount of training will assure that all edge cases that come up in real life would be covered [2]. We believe two key issues must be addressed before such AI systems can be assured: (1) fault tolerance and (2) ML competence.

Johns Hopkins Institute for Assured Autonomy

The complicated nature and scale of these systems make them infeasible to model with high fidelity to provide strong design-time certification. Traditional Simplex-based approaches to provide fault tolerance to safety critical systems may not be sufficient. These approaches can either be too lax and allow the system to enter unsafe states or encumber the system such that no performance can be gained from autonomy. However, some method of fault tolerance is required since edge cases and adversarial inputs seem inherent in many AI based solutions.

We introduce RADICS: Runtime Assurance of Distributed Intelligent Control Systems, to help solve some of these problems. RADICS uses both black and white box monitoring in a Simplex-like [3] approach to create a reliable system that achieves good performance on average, without suffering from failure cases as straight AI systems do. A decision module takes input from both monitors to determine the correct action.

The black-box ensures correctness by detecting when the system is on a failure trajectory and switches to a provably safe, but less effective algorithm. When the safe algorithm has righted the system, control is given back to the AI controller.

The white-box monitor helps improve the performance by predicting when the system might begin a failure trajectory. The white-box monitor can detect when the AI controller is unsure of the correct action. This indicates it might be worth switching to the safe algorithm sooner, to avoid paying the full cost associated with declining performance to the black-box threshold.

RADICS can only solve problems that have a few preconditions. The first is that no single decision can lead to a total system failure. If any single decision can lead to failure, then RADICS would always have to let the safe algorithm control the system, defeating its purpose. This likely eliminates some applications such as self-driving cars, but still allows for many other system control problems where the system can gradually fall into bad states.

The second is that a safe alternative algorithm is known. If the safe algorithm is provably correct, RADICS can ensure the whole system is safe. However, even if we cannot find a provably good algorithm, RADICS can still be useful. If some algorithm is currently being used to solve the problem and has been deemed acceptable, then RADICS can achieve performance at least as good in the worst case, and normally better, failing, at most, as often as the acceptable algorithm.

978-1-6654-3950-3/21/$31.00 ©2021 IEEE
DOI 10.1109/DSN-W52860.2021.00038

Contributions

1) We introduce the RADICS architecture, the first to combine black and white box monitoring to maximize the performance of assured AI systems.
2) We present a rudimentary traffic control case study to show the effectiveness of the RADICS architecture.

II. RADICS Architecture

RADICS is an architecture for creating high accuracy, dependable AI systems by combining highly accurate AI techniques with monitors to ensure correctness. RADICS uses both black and white box monitoring to maintain high accuracy, while ensuring correctness. We first describe how to use black-box monitoring and then extend this to take advantage of white-box monitoring.

A. Black-Box Monitoring

Black-box monitoring is a standard approach for creating reliable systems. A black-box monitoring system involves four major components: a safe controller, which is able to control the situation in an acceptable manner; an untrustworthy controller, with better average performance, but may suffer from unacceptable faults; a monitor, which looks at the state of the entire system and determines if the system is in a good state; and a decision module, which chooses which controller to use at any point in time.

a) Safe Controller: The safe controller is a system component fully capable of controlling the system in any state. The safe controller can be a simple, static algorithm, which has theoretical guarantees about its performance. However, this safety often comes at a cost and thus the safe controller is expected to have worse performance in many cases. In situations where provably good safe controllers are hard to create, the need for them can be alleviated with a small amount of risk. If the problem is currently being solved, then there is some solution which has an acceptable level of risk. This solution can be used instead of a provably safe controller and RADICS will allow the overall system performance to increase, while still only failing in the situations where the existing solution would fail.

b) AI controller: The AI controller is a component also capable of controlling the overall system. It should, on average, perform better than the safe controller. However, the AI system may incur unacceptable faults. AI systems are expected to be able to perform much better on the common case, but current research indicates that there will always be edge cases or adversarial scenarios that exist and cannot be eliminated by simply training more, thus the need for a higher level system such as RADICS.

c) Black-Box Monitor: The black-box monitor looks at the overall state of the system and determines how far the system is from breaking any invariant. This is equivalent to saying which state from Figure 1b we are in.

d) Decision Module: The decision module is responsible for determining when the system should switch between controllers. It chooses the safe controller whenever it needs to so that it can ensure correctness and the AI controller to improve overall performance since the AI controller is expected to outperform the safe controller in common situations.

The system, which can be seen in Figure 1a, roughly works as follows: State is collected from the environment and sent to two controllers, one known or proven to be safe and an AI controller, which is expected to give good performance. State is also sent to a black-box monitor, which can detect if the system is in danger. Each controller proposes an action and sends it to the decision module. The monitor sends information about how far the system is from breaking invariants. The decision module uses this distance and the state information to choose between actions. When the system is in the `Safe Region`, as shown in Figure 1b, the AI can control the system since there is no possibility of failure. When the system enters the `Danger Region`, the decision monitor selects the safe controller, so the system has time to right itself before it reaches the `Failed Region`.

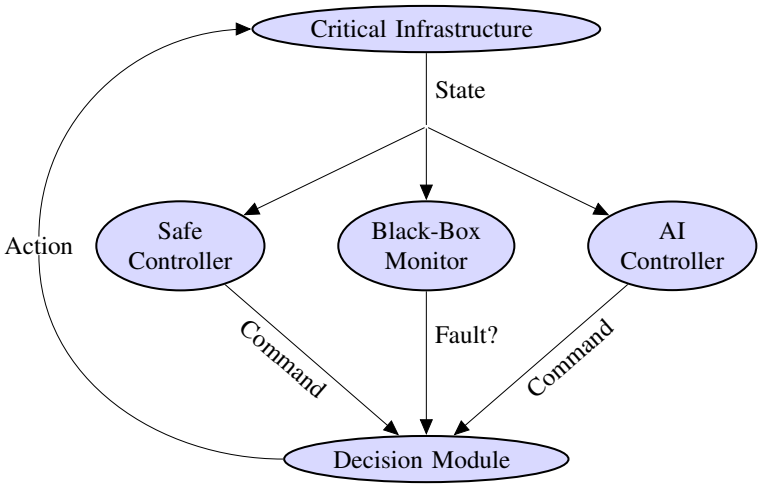
The decision to switch to the safe algorithm from the AI controller is straightforward. It takes some amount of time for the safe controller to right the system after being switched to. We can determine how far it can continue to degrade after switching to the safe controller. Whenever we are within this range, we switch to the safe controller so that it always has enough time to right the system before it fails. This range is represented by the red region in Figure 1b.

The decision to switch back is similar. Whenever the system is far enough from the failure zone (outside the red zone), the system switches to the AI controller. This switching approach will perform correctly in all cases, but can perform poorly. One such case is if the system is in a long-term state which the AI has not trained properly for. This case causes oscillation as follows:

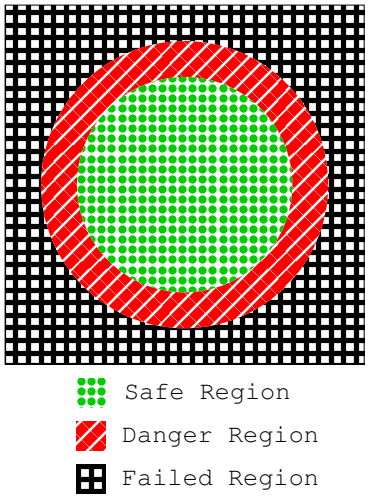
1) The environment is in a state the AI is not trained for
2) As system performance degrades, the system switches to the safe controller
3) Once the performance has stabilized, the system switches back to the AI controller
4) The AI controller still fails at this case and thus performance falls until it switches back to the safe controller

This oscillation can be handled by using simple timers. After switching to the safe controller, the decision module requires that it stays there for some amount of time. We introduce a white-box monitor, which enables more complex switching mechanisms, which can help eliminate oscillation.

Black-box monitoring has previously been used for AI systems [4], [5]. However, only using a black-box monitor limits the types of switching mechanisms used. Sometimes these systems are overly conservative, such as in [3], [6], where once the system has switched to a safe controller, it remains there until reset manually. Another approach uses reachability analysis to determine that it will stay in the `Safe Region` for at least some amount of steps [4], [5], but this



(a) Black-box monitoring system diagram. System state is collected from the environment and sent to the two controllers and the black-box monitor. Each controller issues a command and sends it to the decision module. The decision module takes these two commands along with a fault distance from the monitor to create an action.



(b) The AI controller influences behavior when any invariant is not in danger of being violated and thus is in the `Safe Region`. The monitor determines whether the system is in the `Danger Region` and then the safe controller takes over to avoid the possibility of reaching the `Failed Region`.

Fig. 1: Basic black-box monitoring overview

can lead to oscillations as shown above unless timers or such are used.

B. White-box Monitoring

One of the problems with black-box monitoring is that the system does not know anything is wrong until we are already in a bad state. This, along with the fact that it can take some amount of time for the safe controller to right the system, causes a dip in performance whenever the AI controller is incapable of handling the situation. We can help alleviate some of these issues with white-box monitoring, which can look into the state of the AI controller, and thus have a better idea of how it is behaving. The general idea is that the white-box monitor will be able to predict when the AI controller is likely to make a mistake and switch to the safe controller earlier, as the performance starts to degrade, before the black-box monitor can detect anything is amiss.

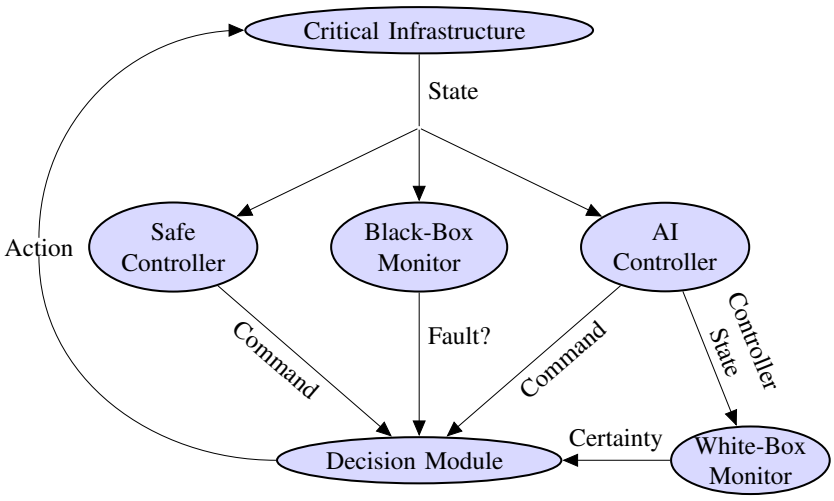
The main task of the white-box monitor is to determine how confident the AI controller is in its decision. If the white-box monitor can determine that the AI controller is not very confident in its actions, we can switch to the safe controller before a large drop in performance happens. We can think of this in the same fashion as the black-box monitor with the addition of a `Questionable Region` (Figure 2b) to the existing `Safe Region`, `Danger Region`, and `Failed Region`. This works as follows: if we are getting close to the area that the black-box monitor would have to save us, we might want to switch away from the AI controller to avoid a drop in performance. However, if the white-box monitor determines the AI controller has high confidence in its action, we will allow it to go up to the `Danger Region` region, while if it detects low confidence, we will switch it sooner. The higher the confidence, the farther into the `Questionable Region` region the decision module will allow the system to progress before switching to the safe controller.

The white-box monitor can also be used when switching back to the AI controller from the safe controller, when the system has reached the `Questionable Region`. The white-box monitor determines how confident the AI controller is, and its actions are only taken if it is very confident. This can help eliminate the oscillation issue described above since the white-box monitor can determine if we are likely to perform poorly when we switch back, which can prevent the unnecessary switches away from and back to the safe controller.

The white-box monitor alone is not sufficient to keep the system safe. The white-box monitor can only determine the AI controller's own expected performance, given the current situation. If the situation is rapidly changing, or the white-box monitor does not assess the situation correctly, the black-box monitor is still necessary to ensure overall system correctness.

III. Case Study: Reinforcement Learning for Traffic Lights

We examine the effectiveness of RADICS on a case study of traffic light controllers. We use the Simulation of Urban Mobility (SUMO) framework [7] to simulate the traffic light system. Each simulation is a two-by-two traffic grid. We define outside edges as edges on which vehicles enter the grid. There are eight outside edges for the grid, as seen in Figure 3. Vehicles are routed randomly. Each simulation step



(a) RADICS System diagram. In addition to the black-box monitoring system in Figure 1a, we add a white-box monitor which receives state from the AI controller and determines the AI's certainty in its decision. The decision modules uses this extra information to improve the performance of the system.

(b) When the system is in the `Questionable Region`, the decision monitor uses the output from the white-box monitor to select which controller to use. This can allow us to switch away from the AI controller sooner when it is unsure and likely to have bad results

Fig. 2: RADICS overview

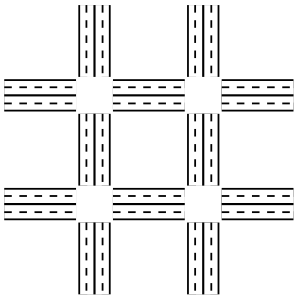


Fig. 3: Example Traffic Grid

is equivalent to 0.1 seconds in real time. The task is to control the lights to maximize the throughput of the system.

This is a fairly simple task, which reinforcement learning methods have been used on before to improve system behavior [8]. The concern with using any AI solution in infrastructure is what will it do in untrained scenarios. We will describe a simple AI solution to this task and then show one such scenario where the AI system fails, and the simple, safe controller performs well.

We implemented the four components of the black-box approach as described in Section II-A and a rudimentary white-box monitor.

The safe controller is a simple, timer-based approach. The light phases[1] go in a cycle timed so that each car will wait, at most, once when attempting to go through an intersection. This approach performs reasonably well in all situations.

We implement a deep reinforcement learning AI controller using the Flow framework [9]. The model was trained on 500 vehicles per hour on each outside edge for 80 million steps[2]. After the training, the model performed around 10% better than the safe controller on the trained scenario. We note that the model was insufficiently trained, but as [2] discusses, all AI systems are expected to have untrained edge cases.

The black-box monitor is a simple mechanism which tracks the average speed of all cars in the system. We want to prevent the system's average speed from dropping too much.

The white-box monitor, on the other hand, runs a test simulation every 100 steps. It uses the average inflow rates of the past 100 steps to simulate the next 1000 steps and outputs the average speed.

The decision module takes in the average speeds from both the black-box and white-box monitors. Switching to the safe controller happens whenever the black-box monitor detects the speed has dropped below 4 m/s or the white-box monitor predicts that the system will in the near future. We switch back when both the black-box measures the current average speed to be above 5 m/s and the white-box monitor predicts the average speed to stay above 5 m/s assuming the near past represents the near future. To avoid oscillation, we wait at least 60 seconds after switching to the safe controller before we can switch back to the AI controller.

[1]A light phase describes which lanes are allowed to proceed through the intersections

[2]In our environment this training took about a week.

TABLE I: Average speed of each controller in meters per second. Segments 1 and 3 are the trained scenarios, while Segment 2 is an anomalous scenario. We see that during segment 1, all AI based controllers perform the same and outperform the safe controller. During the anomaly in segment 2, the AI crashes, but RADICS is able to rescue the system from performing too poorly, just using a black-box monitor is able to control the crash, but not as well.

Controller	Overall	Segment 1	Segment 2	Segment 3
Safe controller	5.65	5.61	5.90	5.63
AI Controller	5.53	6.23	3.42	5.47
Black-Box	5.76	6.23	4.66	5.64
RADICS	5.94	6.23	5.06	5.91

IV. Evaluation

We present a preliminary evaluation of RADICS on the traffic grid problem. We evaluate four different controllers: the safe controller, the AI controller, a black-box monitoring approach, and RADICS with both black and white box monitoring.

We test the system on two different scenarios. The first is the trained scenario: 500 vehicles per hour on each edge. The second is an anomalous scenario that we found that the AI controller performs poorly on. Specifically, we have an inflow of 500 vehicles per hour on one edge and 100 vehicles per hour on the other edges. We find this scenario very interesting because we would never expect the AI to perform worse, because the scenario simply removes cars from the trained scenario, so one would expect that running a similar algorithm should work at least as well.

The evaluation runs as follows: The system starts with the trained scenario, since we expect the AI system to be well suited for the common case; we call this part segment 1. Then, at some point, we switch the system to a case we know the AI controller will perform poorly on, which we call segment 2. After a brief time, the system goes back to the trained scenario, which we call segment 3. An ideal system will exceed the safe controller in both the first and last segments and perform similarly in the middle case. Specifically, we ran a simulation of 25,000 steps; segment 1 has a duration of 10,000 steps, segment 2 has duration 3000 steps, and segment 3 has duration 12,000 steps. Since each step represents 0.1 seconds, this gives us 5 minutes of time when the system deals with the anomaly and 20 minutes to observe how the system corrects itself afterwords.

We present the full results in Table I. The AI controllers perform better than the safe controller in the regular scenario. Once the system enters the anomalous scenario, the safe controller outperforms the AI based controllers. When the system reverts to the trained scenario, both monitoring based systems switch to the AI controller and give better results. The AI controller alone is still in a bad state in Segment 3 because it takes a substantial amount of time to recover from the anomalous period.

To show how the system behaves during the anomalous scenario in more details, we look at the average speed for each controller over time in Figure 4. Additionally, we normalize the average speed of AI-based controllers with respect to the safe controller in Figure 5. We see that in Segment 1, all AI based systems run identically, since they all run the same AI controller. We notice that the AI controller outperforms the safe controller. When Segment 2 begins, we see all AI based controllers dip in performance; however, both monitoring based ones are able to catch the drop in performance and rescue the system. The addition of the white-box monitor in RADICS is able to catch the anomaly sooner, and as such, performs better. The simple AI controller cannot handle the untrained scenario — the system crashes and recovers slowly when the system returns to a trained scenario.

During Segment 2, the RADICS controller with both monitors switches to the safe controller much earlier than the one with only a black-box monitor. This is because the white-box monitor successfully detects the anomaly based on the inflow from the near past sooner than the black-box monitor was able to detect it. With only the black-box monitor, the RADICS controller oscillates between the AI and safe controllers since the system decides to switch to the AI controller each time the safe controller brings the performance up. On the other hand, RADICS with both monitors determines that the system is still in the anomalous case by constantly running test simulations and then stays with the safe controller.

V. Conclusion

We have introduced the RADICS framework for creating reliable and performant systems to enable the use of AI in critical infrastructure. We used black-box monitoring to ensure system correctness, while using white-box monitoring to minimize the cost of the AI component failures and thus maximize the overall system performance. We implemented a version of RADICS to manage the traffic lights in an intelligent traffic control system for a small grid and showed the effectiveness of RADICS in its ability to improve the average traffic speed.

Future work mostly involves creating a larger and more complicated case study to fully show the effectiveness of RADICS. To do this, we will both use a larger and more complicated traffic grid which will involve the use of scalable ML techniques, rather than the monolithic approach currently used, and continue research into better white-box monitors. The current white-box monitor relies entirely on simulation; we envision different ones which are able to determine such things as how similar the current situation is from ones that we trained on and how far the current state is from making a different decision. Ideally, multiple measurements would be used jointly to create a robust white-box monitor.

Lastly, we hope to look into the problem of composing multiple RADICS protected infrastructure as part of a larger, reliable system. For example, each traffic light would have its own RADICS, and they can switch between states, both individually and jointly, to maximize system performance.

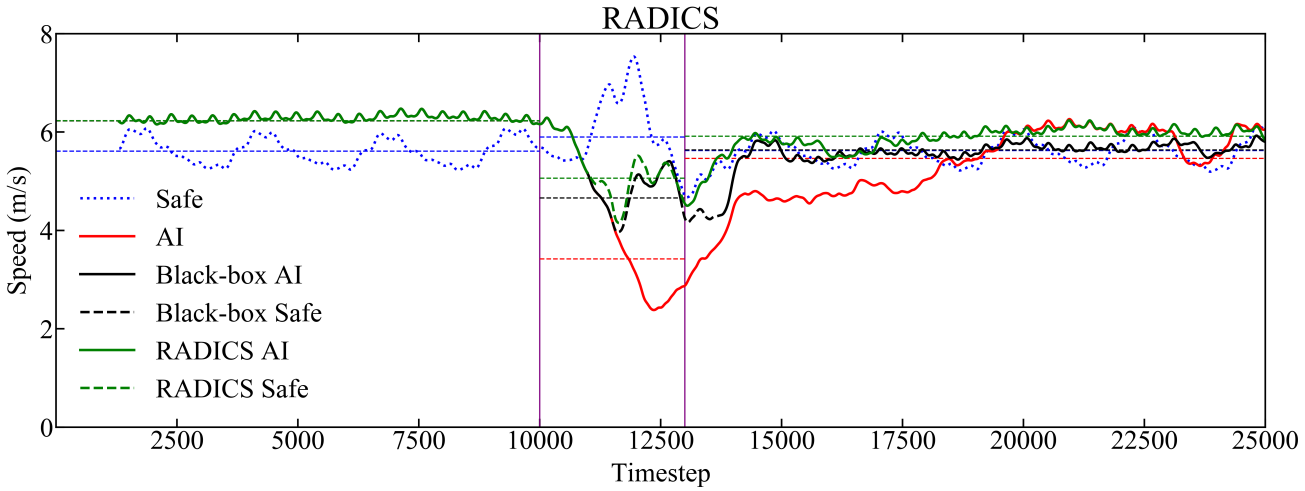


Fig. 4: The average speed of all vehicles in the system over time. We evaluate four different controllers: the safe controller, the AI controller, a black-box monitoring approach, and RADICS with both black and white box monitoring. Horizontal lines show the average speed of each controller in each segment, whereas vertical lines mark the start and end of the anomalous scenario. We use the dotted line when a safe controller is in control and the solid line when an AI controller is.

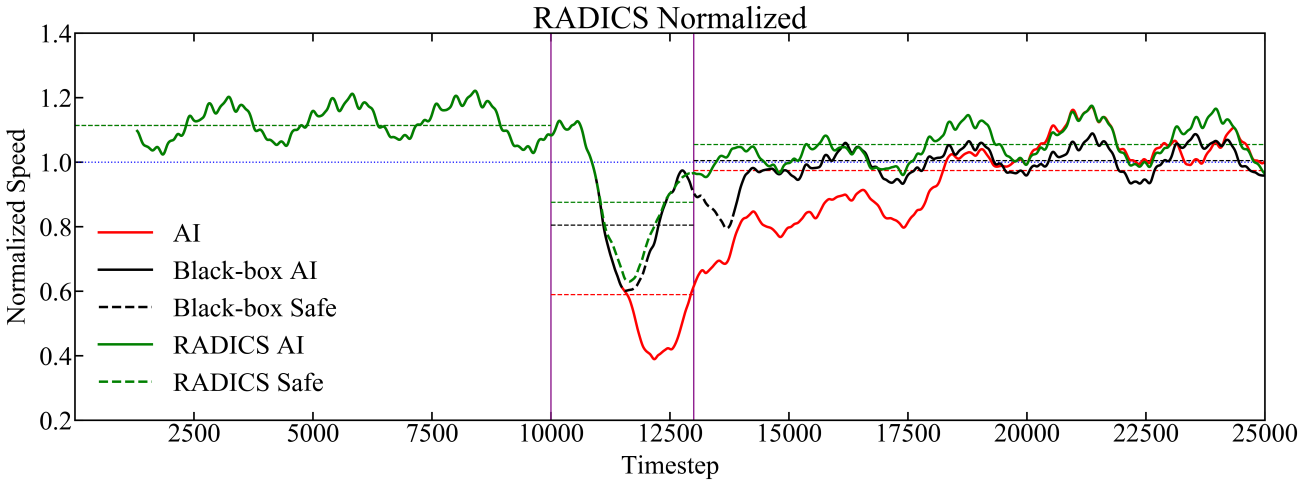


Fig. 5: Normalized average speed of AI-based controllers with respect to the safe controller. We divided the average speeds of AI-based controllers in Figure 4 by the average speed of the safe controller.

References

[1] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 427–436, 2015.

[2] A. Shafahi, W. R. Huang, C. Studer, S. Feizi, and T. Goldstein, "Are adversarial examples inevitable?," *arXiv preprint arXiv:1809.02104*, 2018.

[3] D. Seto, B. Krogh, L. Sha, and A. Chutinan, "The simplex architecture for safe online control system upgrades," in *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No. 98CH36207)*, vol. 6, pp. 3504–3508, IEEE, 1998.

[4] D. T. Phan, R. Grosu, N. Jansen, N. Paoletti, S. A. Smolka, and S. D. Stoller, "Neural simplex architecture," in *NASA Formal Methods Symposium*, pp. 97–114, Springer, 2020.

[5] A. Desai, S. Ghosh, S. A. Seshia, N. Shankar, and A. Tiwari, "Soter: a runtime assurance framework for programming safe robotics systems," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 138–150, IEEE, 2019.

[6] D. Seto and L. Sha, "A case study on analytical analysis of the inverted pendulum real-time control system," tech. rep., CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 1999.

[7] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*, IEEE, 2018.

[8] M. A. Wiering, "Multi-agent reinforcement learning for traffic light control," in *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*, pp. 1151–1158, 2000.

[9] C. Wu, A. Kreidieh, K. Parvate, E. Vinitsky, and A. M. Bayen, "Flow: Architecture and benchmarking for reinforcement learning in traffic control," *arXiv preprint arXiv:1710.05465*, p. 10, 2017.