Dependable AI for Traffic Light Control Systems

by

Jerry Chen

A project report submitted to Johns Hopkins University
in conformity with the requirements for the degree of
Master of Science in Engineering

Baltimore, Maryland
December, 2021
Advisors: Dr. Amir Yair, Brian Wheatman

Acknowledgments

I would like to thank Brian Wheatman for his contributions to this work, along with his continued support and guidance. Brian started this project and had a vision on how to design different types of white-box monitors and the white-box decision module, leading to the full RADICS architecture we have today. He is also very knowledgeable in performance engineering and helped reduce some overhead in SUMO.

I would also like to express my gratitude to my advisor, Dr. Yair Amir, who helped me in countless ways throughout my last two years at Johns Hopkins University. He was my professor in Distributed Systems and Advanced Distributed Systems and Networks. I've been working as his teaching assistants for the past three semesters. His passion on helping students and contributing to the society truly inspired me.

Additionally, I would like to thank Sahiti Bommareddy, who helped setup our simulation environment. She also offered tremendous help with my projects when I was taking the Distributed Systems course.

This work is supported in part by the Johns Hopkins Institute for Assured Autonomy.

Table of Contents

1 Introduction

Traffic has become a serious issue as more people are moving to metro areas. From a report by the Texas Transportation Institute, drivers in the U.S. spend around 42 hours in traffic every year, which wastes over 3 billion gallons of fuel [1]. Not only does traffic jam wastes time and resources, it also pollutes the environment. According to the United States Population Fund [2] and the Population Reference Bureau [3], these problems may get worse in the future because the population is growing and moving to urban areas in many countries. The current traffic light infrastructure is obsolete and cannot meet the demand of modern cities. We need a smart traffic light system.

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. An AI-based traffic light can reduce driving time and save resources. In this report, we present a Reinforcement Learning (RL) based intelligent traffic light control system.

AI-based systems also come with a major limitation: it is difficult to understand how exactly these systems work and predict when and how they will fail. In the vast majority of cases, these systems outperform traditional systems. However, in edge cases, they fail in unexpected ways [4].

To leverage the exceptional performance brought by AI-based systems while keeping the system safe, we introduce RADICS: Runtime Assurance of Distributed Intelligent Control System, which uses black and white box monitoring to create a dependable system that achieves good performance on average, without suffering from failures caused by edge cases.

A black-box monitor ensures the correctness of the system by detecting when the system performance is on a failure trajectory and rescues the system by switching to a safe controller that guarantees an acceptable level of performance. The AI controller takes over the control of the system when the safe controller has righted the system.

A white-box monitor help improve the performance by predicting when the system might begin a failure trajectory. The white-box monitor can measure the confidence level of the AI controller. If the AI controller is uncertain

about the correct action, 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.

We evaluate RADICS on different adverse traffic scenarios and show how one can use RADICS to construct dependable AI systems. With both black and white box monitoring, RADICS is able to protect the system from crashes during adverse events while still allowing for the AI controller to improve the overall system performance.

2 Related Work

Traffic light systems have been computerized after the invention of the computer in 1960s. By monitoring traffic and running optimization algorithms on computers, we develop better traffic light algorithms over time [1].

Google is among the first to bring a smart traffic light system to the real world. After a successful implementation of the AI-based traffic light system in Israel, Google claims that the new system cuts the fuel consumption and traffic delays at intersections by 10% to 20% , with a concern that letting AI make all decisions could potentially be risky [5].

Several approaches have been proposed to improve dependability in AI-based systems, such as using Baysian inference and model ensembles [6]. Additionally, new classifiers, such as uncertainty-based OOD-classifier (UBOOD), can be used to detect an AI model's uncertainty when encountering out-of-distribution (OOD) data [6]. In this report, one of the white-box monitors we introduce is similar to the model ensembles method, but instead of using different model architectures, we train multiple models with varying randomization.

Black-box monitoring is also commonly used when developing dependable AI systems [7]. For instance, Soter, a robotics programming framework, includes a black-box based integrated runtime assurance (RTA) system that allows the use of uncertified components, while still providing safety guarantees [8]. However, only using a black-box monitor limits the types of switching mechanisms that can be used. Sometimes, these systems are overly conservative, such as in [9] and [10], where once the system has switched to a safe controller, it remains there until reset manually. Our work aims to develop methods in addition to black-box monitoring to make AI systems dependable while still benefiting from its performance.

The Simulation of Urban Mobility (SUMO) is an open source traffic simulation framework mainly developed by employees of the Institute of Transportation Systems at the German Aerospace Center [11]. SUMO is highly cuztomizable. It allows users to define traffic patterns, driver behaviors, traffic light algorithms, roads and lanes. SUMO also provides APIs to control traffic lights at runtime. We use SUMO to create our traffic simulation environment.

3 Simulation

We create a realistic traffic simulation using SUMO. In our simulation, we define a two-by-two grid with 2 lanes on each road, on which vehicles can turn left or go straight on the left lane and the right lane is right turn only. We define outside edges as edges on which vehicles enter the grid. As shown in Figure 1, there are 8 outside edges in the grid. Inflow vehicles have a $\frac{1}{6}$ chance of turning either direction at each intersection. The rest of the vehicles will go straight and exit the grid. Each vehicle can turn at most once.

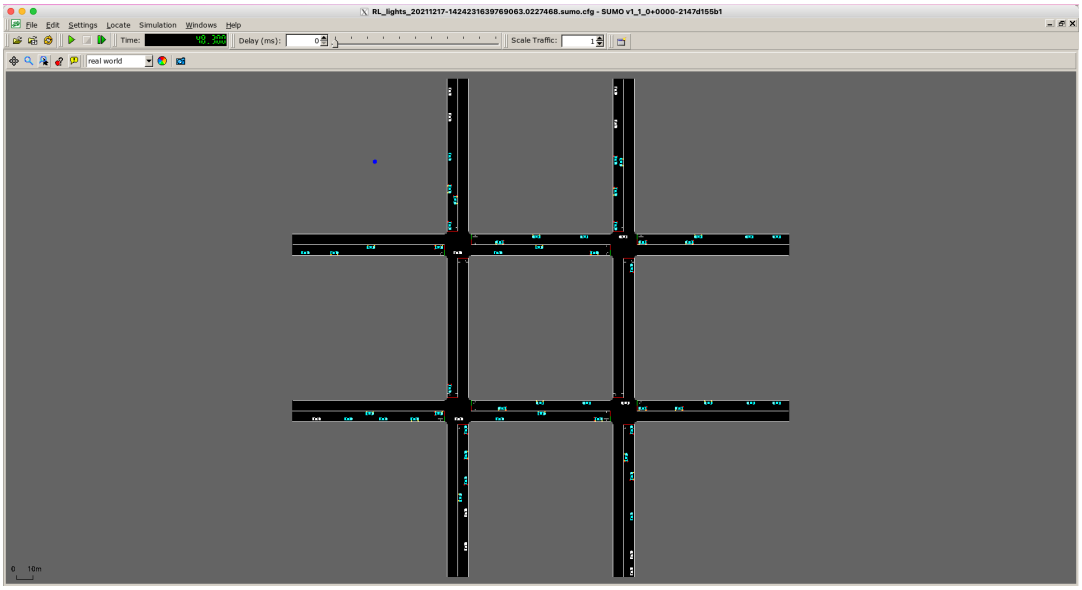

Figure 1: Traffic Simulation

Occasionally, two vehicles can collide with one another at an intersection to simulate car crashes in real life. We teleport these vehicles to their destination roads as if they went through the intersection smoothly. Drivers are allowed to speed up when going through an intersection and use the emergency brake when needed.

Each simulation step is equivalent to 0.1 seconds in real time. We define the traffic light algorithm as follows: green light for 26 seconds, yellow light for 3 seconds, left-turning green light for 5 seconds, yellow light for 3 seconds, red light for 26 seconds, yellow light for 3 seconds. All traffic lights are synchronized to create a flow, as seen in Figure 2, so that vehicles can travel from one side of the grid to the other without stopping or slowing down.

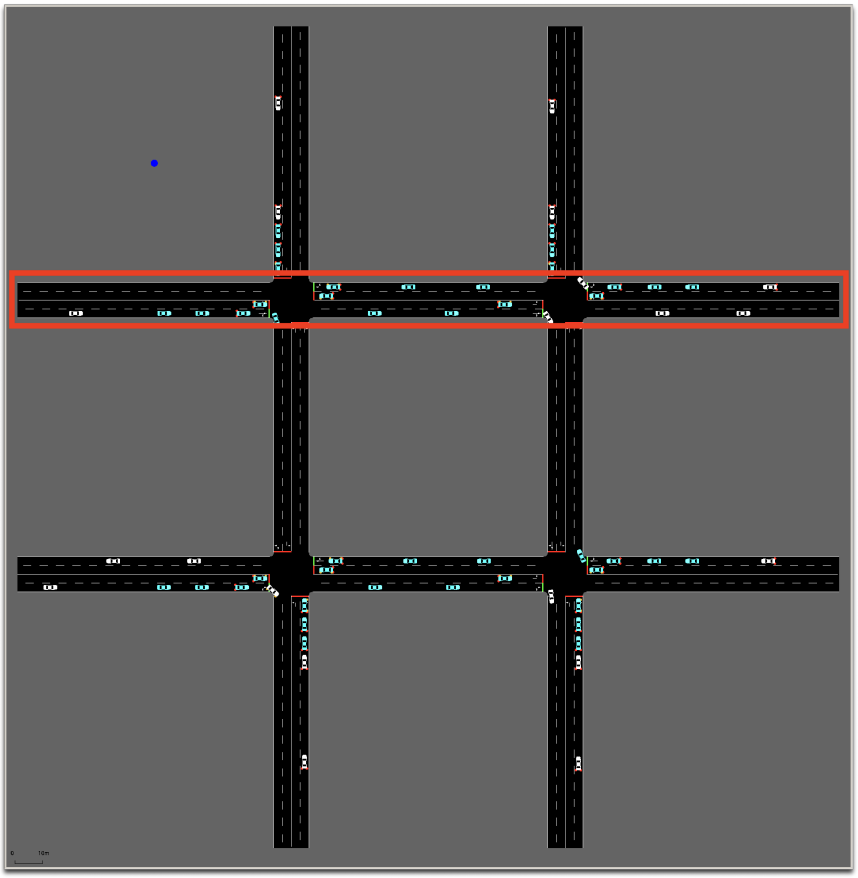

Figure 2: Vehicle Flow

This static traffic light algorithm can guarantee that any vehicle that is going straight will stop at most once at an intersection, even when there is an overwhelming amount of cars in the grid.

4 Reinforcement Learning

We implement a deep reinforcement learning (RL) AI traffic light controller using the FLOW framework [12], which is built on top of TensorFlow, Stable Baselines, and SUMO [11]. We use Proximal Policy Optimization (PPO) [13] algorithm to train RL models because PPO performs comparably or beter than the state-of-the-art RL algorithms and it is simple to implement and tune [14]. Specifically, we use the PPO2 model from Stable Baselines, which is an implementation of PPO using OpenAI gym. It uses vectorized environments for mutiprocessing.

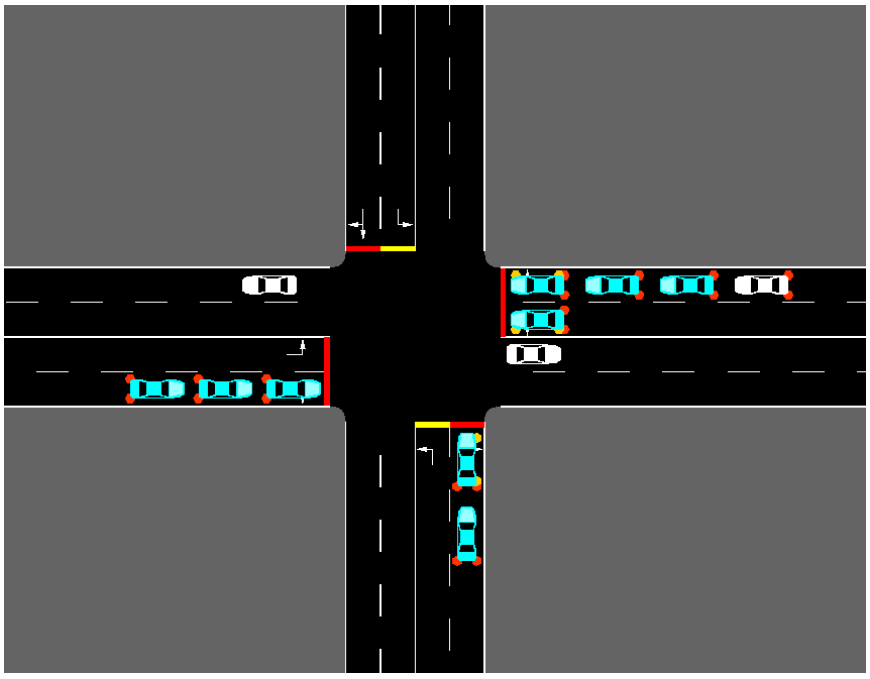

Figure 3: RL Observations

We take a monolithic approach when training the AI. In our single-agent RL setup, we pass the information of vehicles, traffic lights, and traffic condition on each edge, to the RL agent. The following describes the full details of the feature vector:

- Current speeds of observed vehicles[1] in the grid

[1]We define observed vehicles as the top three vehicles that are closest to an intersection

- Distance to the intersection that a vehicle is heading to for all observed vehicles
- IDs of edges which each observed vehicle is on.
- Information about density of cars for each edge
- Average speed of vehicles of each edge
- Number of time steps since last time a light has changed for all traffic lights
- The direction in which a light is allowing cars to go for all traffic lights
- Whether a light is currently yellow for all traffic lights

The model was trained with an inflow of 500 vehicles per hour on each outside edge for 80 million simulation steps. In our simulations, arrived vehicles are defined as vehicles that reach their destinations and exit the grid. After every 3000 time steps, we calculate the average speed of all vehicles in the system, reset the simulation environment, and use the following reward function to update the model:

$$\text{Average speed of all vehicles} + 1.67 \times \text{Number of arrived vehicles}$$

Table 1: Average speed of all vehicles in a simulation of 3000 steps when using a model trained with different duration.

Steps	Average Speed (m/s)
50,000	1.98
10,000,000	3.02
20,000,000	3.75
30,000,000	5.66
40,000,000	5.54
50,000,000	6.00
60,000,000	6.35
70,000,000	6.37
80,000,000	6.46

they are heading to, on each lane. These vehicles are marked blue in Figure 3. Thus, there can be at most 6 observed vehicles on each edge.

Table 1 shows the training progress over time. The improvement of the RL model is significant. We used a learning rate of 5×10^{-4} initially. Then we changed the learning rate to 1×10^{-4} after the performance stops to improve at 40 million steps. After 80 million steps, the model outperforms the static traffic light algorithm by 10% to 20%.

5 RADICS

RADICS is an architecture for creating high efficient, 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 in a standard simplex-like approach and then extend this to take advantage of white-box monitoring. Lastly, we present the complete RADICS architecture in the context of traffic simulation.

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, which has 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.

The safe controller is fully capable of controlling the system in any state. It 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 on average. 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.

The AI controller is also capable of controlling the overall system. It should, on average, outperform 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 cases, but current research indicates that there will always be edge cases or adversarial scenarios that exist and cannot be eliminated by more training, thus the need for a higher level system such as RADICS [15].

The black-box monitor observes the overall state of the system determines how far the system is from breaking any invariant. To ensure correctness, the black-box monitor must always be able to determine when the system is within some distance from any failure state. This is equivalent to saying which state from Figure 4 we are in.

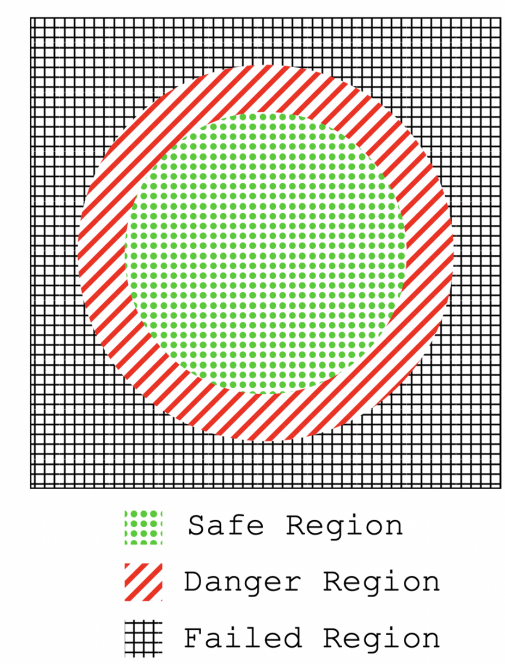

Figure 4: Black-box monitoring decision module

The decision module is responsible for determining when the system should switch between controllers based on the output of the black-box monitor, as explained in Figure 5. It chooses to switch to the safe controller when the system is in the `Danger Region` so that it can ensure correctness. To improve overall performance, it switches the system to an AI controller when it determines that the system is in the `Safe Region`.

One inherent limitation with black-box monitoring is that it can cause oscillations when the system is under a scenario which the AI controller cannot handle. As the system performance degrades and enters the `Danger Region`, the black-box decision module will switch to the safe controller, which recovers the performance and brings the state back to the `Safe Region`. However, the scenario hasn't changed yet and the decision module will soon switch the system to the safe controller again. Another limitation with black-box monitoring is that it does not recognize the bad situation until we are already

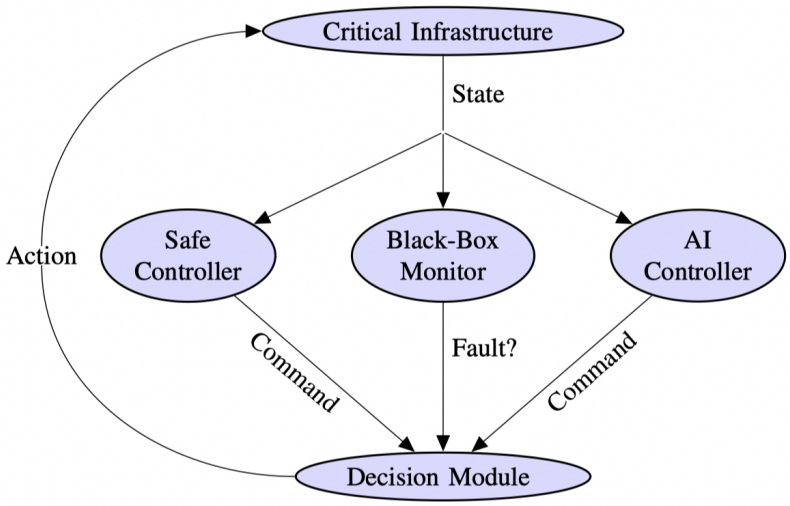

Figure 5: Black-box monitoring diagram

in the `Danger Region`. This performance drop, along with the oscillation issue, can be alleviated by introducing white-box monitoring to the system.

White-box Monitoring

We introduce white-box monitoring, which can look into the state of the AI controller and predict when the AI controller is likely to make a mistake so that it can switch to the safe controller before the black-box monitor can detect anything is wrong.

The main task of the white-box monitor is to determine how confident the AI controller is in its decision. We extend the decision module for white-box monitoring by adding the `Questionable Region`, which can be seen in Figure 6. A white-box monitor will first measure the confidence level of the AI controller. If the AI controller is confident, the system will allow the AI to take control up until the `Danger Region` is reached. The higher the confidence, the further into the `Questionable Region` the decision module will allow the system to progress before switching to the safe controller. This can help mitigate the performance drop caused by not switching to the safe controller early enough.

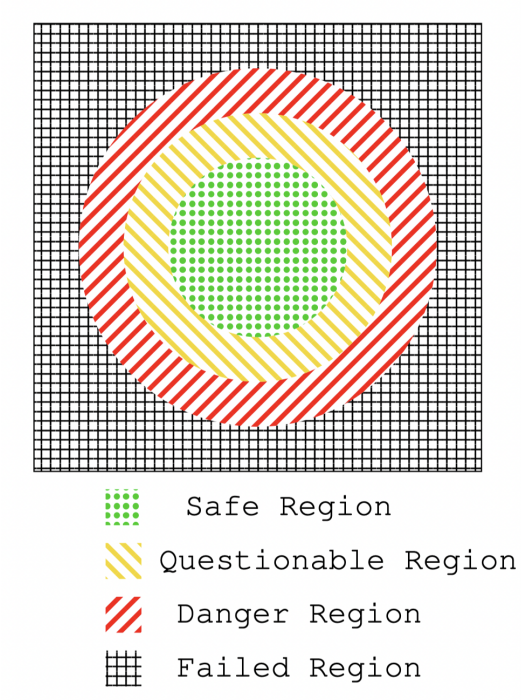

Figure 6: White-box monitoring decision module

To avoid the oscillation issue when switching back to the AI controller, we use the white-box monitor to measure the AI controller's confidence level when the system reaches the `Questionable Region`. We only switch to the AI controller if the confidence score is high.

We describe three types of white-box monitors.The first type uses the information that the AI system already calculates in its prediction. For example, in PPO models, one of the final steps is a score for each possible decision with its likelihood of being chosen. One can look at the magnitude of the score of the chosen decision to determine its confidence, or how likely it was to choose its selected option compared to any other option. If many possible options have similar scores, or the selected option has a low score, it indicates that the AI controller may have low confidence in its decision. This approach is very cheap since it does not require any additional calculation.

Another option is to train multiple models with different randomization. We then compare the results of these models to the result of the AI controller with a given state. If all models agree with the AI controller's decision, then the AI controller has a high confidence and the current state is likely a scenario

that the AI can recognize. The less the models agree with the AI controller, the lower the confidence is. This monitor has a high start-up cost because training multiple models can take a significant amount of time and computational resources, but evaluating them is very cheap.

The third option is a simulation-based white-box monitor, which constantly runs a test simulation of the near past to simulate the near future. We assume that the near past is similar to the near future. If the test simulation yields good performance, then the AI controller has a high confidence score. Similarly, if the test simulation performs poorly with an input from the near past, then one can assume that the same controller will also perform poorly in the future. This monitor is the most costly one because it requires computational resources to run test simulations all the time.

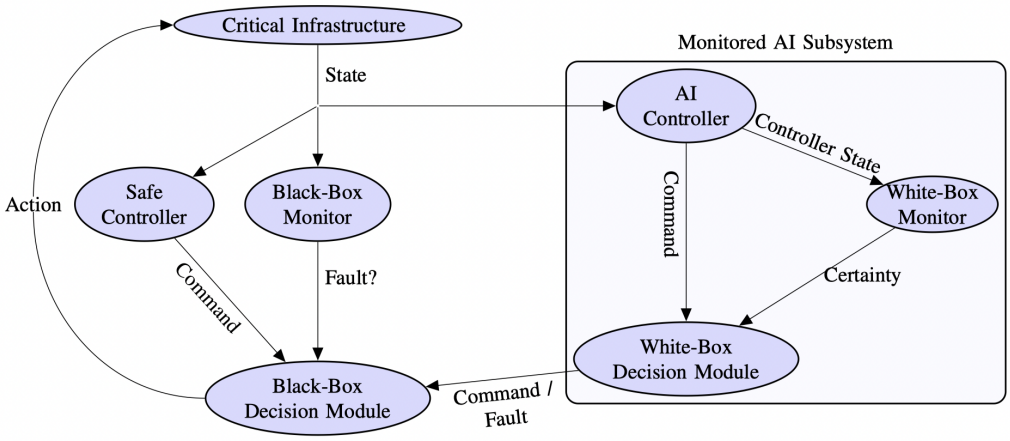

Figure 7: RADICS Architecture

RADICS Architecture

By combining the black and white box monitoring, we present the complete RADICS architecture, as shown in Figure 7. The white-box monitor measures the confidence level of the AI controller and reports it to the white-box decision module while the black-box monitor observes the state of the system. The white-box decision then recommends a decision to the black-box decision module. The black-box decision module then makes a decision upon receiving the black-box monitor's output and the white-box decision module's

suggestion. The black-box monitor's output always takes precedence over the white-box decision module's suggestion when switching to the safe controller so that the safety of the system can be guaranteed. The white-box monitoring can help predict future faults and also alleviate the oscillation concerns.

In the context of traffic simulations, we use the static traffic light algorithm, which is described in Section 3, as the safe controller. The model trained after 80 million steps will be used as the AI controller. We will examine the effectiveness of RADICS in the next section.

6 Evaluation

We evaluate the RADICS approach by making an AI-based traffic light system dependable. We first describe three traffic scenarios that will be used to evaluate the system. Then, we measure and compare the effectiveness of the different white-box monitors in detecting adverse events. Lastly, we present the evaluation of the full RADICS system. We show that without any monitoring, the system performance crashes when encountering an adverse event. Black-box monitoring prevents the full performance impact from the adverse events. The addition of white-box monitors help detect these events sooner and give us better overall system performance.

Traffic Scenarios

Recall that the AI is trained on 500 vehicles per hour on all outside edges. We describe the following adverse scenarios under which the AI controller underperforms:

- **Adverse Scenario 1:** 500 vehicles per hour on all but 1 outside edge which has 100 vehicles per hour.
- **Adverse Scenario 2:** 100 vehicles per hour on all outside edges
- **Adverse Scenario 3:** 500 vehicles on two outside edges on the same side, no cars on any other edge.

Table 2: Average speed of each controller in meters per second in the different scenarios

Controller	Safe Controller	AI Controller
Trained Scenario	5.67	6.27
Adverse Scenario 1	6.17	2.86
Adverse Scenario 2	6.93	5.35
Adverse Scenario 3	5.37	1.34

The performance of both controllers under different scenarios is shown in Table 2. The AI controller's performance crashes in all adverse scenarios

while the safe controller is not affected at all. In the trained scenario, the AI controller outperforms the safe controller by over 10%.

We then define three traffic scenarios that corresponds to the three adverse scenarios. Each traffic scenario starts with the trained scenario for 10,000 steps, after which an adverse scenario runs for 3,000 steps. Then, the trained scenario runs for another 12,000 steps. For example, Traffic Scenario 1 has the Adverse Scenario 1 in the middle. These traffic scenarios are used to evaluate monitors and the full RADICS system.

Monitor Evaluation

Monitors are designed to detect when an adverse event has happened so that the system can switch to the safe controller from the AI controller. The sooner the system can detect this event, the smaller the performance impact from the event is.

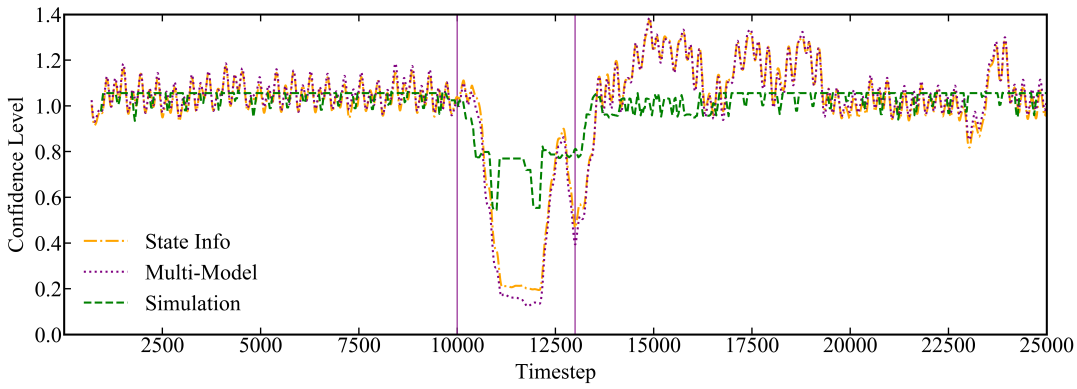

Figure 8: The confidence scores from each white-box monitor around traffic scenario 1, between time steps 10,000 and 13,000; the rest of the time is the trained scenario.

We measure the effectiveness of the white-box monitors by running them with the AI controller the entire time under the Traffic Scenario 1. Figure 8 shows the confidence scores of each monitor, which are normalized to the average of each score to put them on the same scale. All monitors are able to detect when the adverse event has happened. The magnitude of the drop is not important, only that the drop can be easily identified. After the adverse scenario has ended, the confidence scores of these monitors increase to the regular level.

The time it takes for each monitor to detect each of the adverse scenarios is given in Table 3. The sooner the system can identify a problem the sooner the safe controller can take over and rectify the situation. We find that the simulation-based white-box monitor detects the adverse scenarios quicker than the other white-box monitors. However, it is also the most expensive white-box monitor among all three since the system needs to consume computational resources and constantly run test simulations in the background. The state-based monitor is slightly less performant but more cost-friendly than the multi-model white-box monitor - training multiple AI models takes a significant amount of time. All white-box monitors detect the issue much sooner than the black-box monitor.

Table 3: Time after the start of an adverse event when each monitor signals low confidence.

Monitor	Scenario 1	Scenario 2	Scenario 3	Average
Black-Box	1418	969	859	1082
State Information	592	665	566	607
Multi-Model	562	652	467	560
Simulation	406	666	484	518

Full System Evaluation

We present the evaluation of the RADICS system for traffic light control in full details. 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 evaluate the system using the aforementioned traffic scenarios, which are designed to show how well the controller can detect an adverse scenario and protect from the impact of the adverse scenario using the safe controller when necessary, and how they can switch back to the AI controller once the environment returns to a trained scenario.

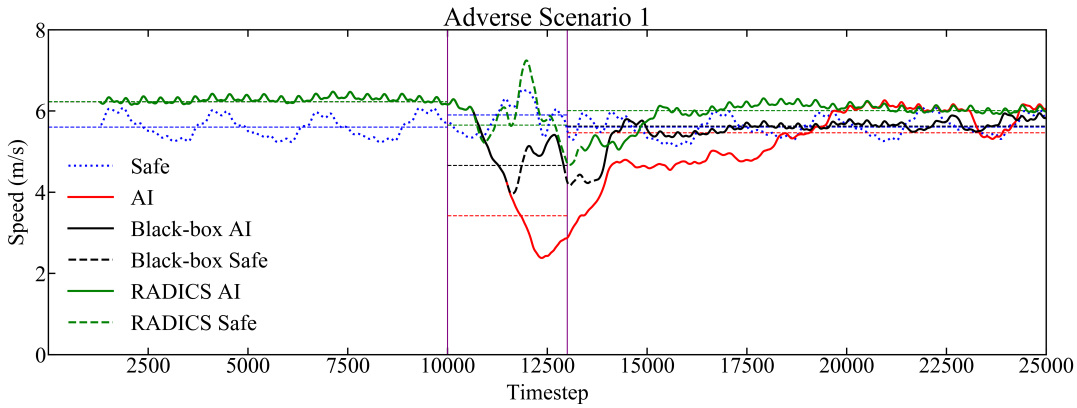

Figure 9: The rolling average of the speed of all vehicles in the system over time under adverse scenario 1 for Segment 2. We evaluate four different controllers: the safe controller, the AI controller, a black-box monitoring approach, and RADICS with both black and simulation-based white-box monitoring. 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.

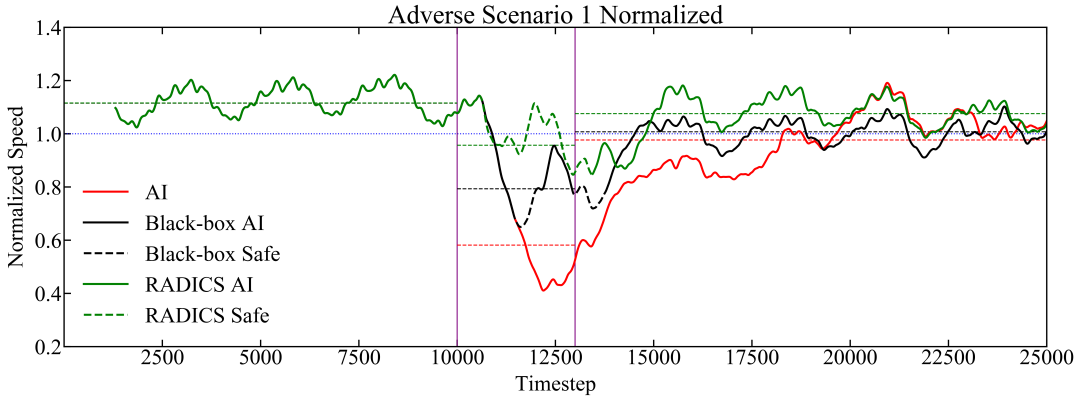

Figure 10: The normalized average speed of AI-based controllers with respect to the safe controller. We divided the average speeds of AI-based controllers in Figure 9 by the average speed of the safe controller.

Figure 9 shows how well each controller performs with traffic scenario 1 when using a simulation-based white-box monitor. Additionally, we normalize the average speed of AI-based controllers with respect to the safe controller in Figure 10. 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 under the trained scenario. 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 simulation-based 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.

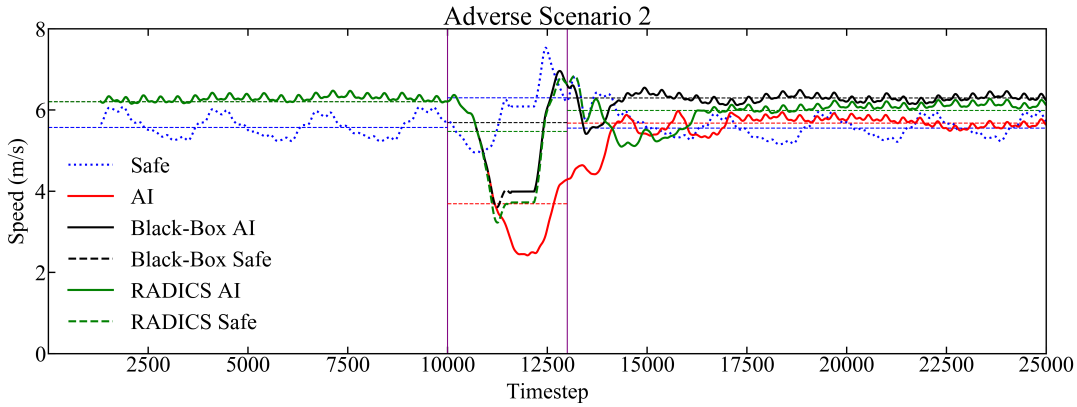

Figure 11: The average speed of all vehicles in the system over time with adverse scenario 2.

Similarly, Figure 11 and 12 show the performance of different controllers with adverse scenario 2 and 3 respectively. In scenario 2 we observe the black-box monitoring approach performing better than the full RADICS approach, this is due to it switching back the the AI controller as soon as the safe controller stopped the fall of the system and the AI managing to perform well in an untrained scenario for a short time. Overall, we still note that in this situation both monitoring approaches significantly outperform just the AI on its own. In traffic scenario 3, we observe that the AI controller is not able to recover after the decline of its performance. The accumulation of vehicles

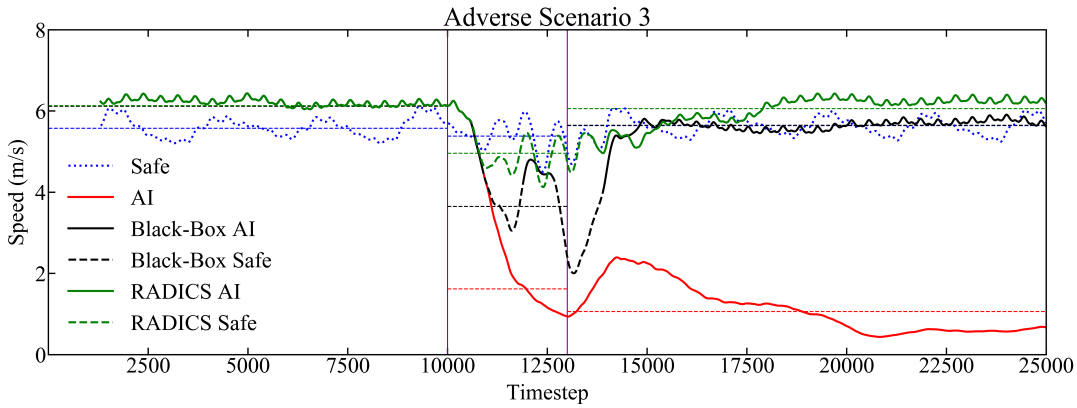

Figure 12: The average speed of all vehicles in the system over time with adverse scenario 3.

pushes the state of the AI controller to the `Failed Region`.

Table 4 shows the evaluation of RADICS under all three traffic scenarios. In adverse scenario 1 and 2, the AI controller has a worse performance in Segment 2 and is able to recover in Segment3. The AI controller crashes in adverse scenario 3, and its performance keep decreasing in Segment 3. In all scenarios, both state-based and multi-model RADICS cannot switch back to the AI controller because the state generated by the safe controller is too different from what the AI controller was trained on. To enable the use of these styles of white-box monitors for the switch back, the AI would need to be specifically trained on situations that arise from running the safe controller so that it can be confident in these situations. Without this specific training, the confidence scores from the state-based and multi-model monitors remain low during Segment 3.

Overall, a system with any monitoring approach performs better than either controller alone. White-box monitors can detect the adverse event sooner than the black-box monitor. Therefore, on average, RADICS systems have a superior overall performance to a system with black-box monitoring only. We also generated simulation videos for all systems in each scenario to visualize their differences in performance.

Table 4: 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. Small differences in Segment 1 are from randomness since some monitoring approaches advance the random number generator. During the anomaly in Segment 2, the AI crashes, but RADICS is able to rescue the system from performing poorly, just using a black-box monitor is able to control the crash, but not as well.

Adverse Scenario 1				
Controller	Overall	Segment 1	Segment 2	Segment 3
Safe controller	5.64	5.60	5.90	5.60
AI Controller	5.53	6.23	3.42	5.47
Black-Box	5.76	6.23	4.66	5.63
RADICS State Info	5.89	6.25	5.74	5.62
RADICS Multi-Model	5.89	6.25	5.73	5.62
RADICS Simulation	6.06	6.23	5.65	6.01
Adverse Scenario 2				
Safe controller	5.66	5.57	6.30	5.55
AI Controller	5.65	6.21	3.69	5.68
Black-Box	6.18	6.21	5.69	6.29
RADICS State Info	5.86	6.20	5.66	5.60
RADICS Multi-Model	5.86	6.20	5.66	5.60
RADICS Simulation	6.01	6.20	5.47	5.99
Adverse Scenario 3				
Safe controller	5.58	5.58	5.38	5.64
AI Controller	3.24	6.11	1.62	1.06
Black-Box	5.59	6.12	3.65	5.65
RADICS State Info	5.75	6.13	4.96	5.62
RADICS Multi-Model	5.75	6.13	5.00	5.62
RADICS Simulation	5.95	6.13	4.96	6.06

7 Conclusion

We have introduced an AI-based traffic light control system. The traffic simulation was developed using SUMO. To construct a more realistic simulation environment, we allowed randomness in driver behaviors, such as vehicles accidents at intersections. Drivers can also choose to change lanes as they wish. We also defined a provably safe traffic light algorithm that guarantees an acceptable level of performance.

We used PPO as the reinforcement learning algorithm to train the AI model with a traffic inflow of 500 vehicles per hour on each input edge. After 80 million simulation steps, the AI achieves exceptional performance. The training took about two weeks in our environment.

We presented two approaches in achieving AI dependability, black and white box monitoring. We introduced three types of white-box monitors that have varying costs and effectiveness. The simulation-based white-box monitor can detect an adverse scenario faster than the other two, but it is also the most expensive monitor because one needs to constantly run test simulations in the background. The multi-model white-box monitor is cheaper but has a high start-up cost. The state-based monitor costs the least since no additional computation is needed to use it. All white-box monitors can detect an adverse event quicker than a black-box monitor. We then combined black and white box monitoring and described the RADICS approach.

Lastly, we defined three adverse scenarios to showcase the effectiveness of three white-box monitors and the RADICS approach. Our evaluation showed that the RADICS approach can achieve better performance than using the safe controller alone, while guaranteeing the safety of the system.

References

[1] J. Palša, L. Vokorokos, E. Chovancová, and M. Chovanec, "Smart cities and the importance of smart traffic lights," in *2019 17th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, pp. 587–592, 2019.

[2] R. Kollodge, B. Crossette, and R. Froseth, "Unfpa: State of world population 2011," 2011.

[3] PRB, "2016 world population data sheet: With a special focus on human needs and sustainable resource," 2016.

[4] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 427–436, 2015.

[5] C. Low, "Google turns its ai on traffic lights to reduce pollution," Oct 2021.

[6] A. Sedlmeier, T. Gabor, T. Phan, L. Belzner, and C. Linnhoff-Popien, "Uncertainty-based out-of-distribution detection in deep reinforcement learning," *CoRR*, vol. abs/1901.02219, 2019.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[14] J. Schulman, "Proximal policy optimization," Sep 2020.

[15] A. Shafahi, W. R. Huang, C. Studer, S. Feizi, and T. Goldstein, "Are adversarial examples inevitable?," *ArXiv*, vol. abs/1809.02104, 2019.