SAVME: Efficient Safety Validation for Autonomous Systems Using Meta-Learning

Marc R. Schlichting[1], Nina V. Boord[2], Anthony L. Corso[1], and Mykel J. Kochenderfer[1,2]

***Abstract*— Discovering potential failures of an autonomous system is important prior to deployment. Falsification-based methods are often used to assess the safety of such systems, but the cost of running many accurate simulation can be high. The validation can be accelerated by identifying critical failure scenarios for the system under test and by reducing the simulation runtime. We propose a Bayesian approach that integrates meta-learning strategies with a multi-armed bandit framework. Our method involves learning distributions over scenario parameters that are prone to triggering failures in the system under test, as well as a distribution over fidelity settings that enable fast and accurate simulations. In the spirit of meta-learning, we also assess whether the learned fidelity settings distribution facilitates faster learning of the scenario parameter distributions for new scenarios. We showcase our methodology using a cutting-edge 3D driving simulator, incorporating 16 fidelity settings for an autonomous vehicle stack that includes camera and lidar sensors. We evaluate various scenarios based on an autonomous vehicle pre-crash typology. As a result, our approach achieves a significant speedup, up to 18 times faster compared to traditional methods that solely rely on a high-fidelity simulator.**

I. Introduction

One way of demonstrating the reliability of an autonomous system is through rigorous real-world testing, a process that requires substantial resources and is often economically infeasible [1]. For this reason, simulations have become the method of choice in both research and industry for the validation of autonomous systems. Compared to real-world testing, simulations are faster, cheaper, and safer [2].

While empirical evidence supports the advantages of simulations over real-world testing, one fundamental question remains: *How much can we trust simulations*? Safety validation relies on a vast number of simulations to find edge cases through falsification [3]. This leads to a dilemma: should the accuracy of the simulation or its runtime be prioritized? Many modern-day simulation tools give the user control over numerous settings that affect behavior, output, and compute cost. Such settings range from different equations of motion to numerical solvers and sensor models [4]. Depending on the system under test, different fidelity settings are more important than others in terms of arriving at accurate safety assessments.

There are two important components of accelerating the validation process: speeding up the runtime and efficiently finding scenarios where the system under test fails. Many different approaches for finding failure scenarios have been studied in the literature. One such approach is black-box optimization [5], which has been applied to safety validation of autonomous systems [6], [7]. Two other similar approaches are path planning [8] and reinforcement learning [9]. A detailed review of methods is provided by Corso et al. [3]. These approaches require a simulation that is assumed to model reality.

Photorealistic simulators such as Carla [10], Airsim [11], or the products by Applied Intuition have been effective tools for developing and testing autonomous driving stacks. This realism, however, comes at the cost of increased computational resources required to run safety analyses. Since most simulators expose a number of fidelity settings to the user, it is possible to adapt those settings for specific needs. For example, an autonomous system stack without cameras does not need rendering. Unfortunately, this tuning process is non-trivial and requires domain expertise. For this reason, there are frameworks that make use of different fidelity settings by combining compute-intensive results from a high-fidelity simulator with computationally cheaper results from a low-fidelity simulator [12]. This approach has been developed further to support more than two levels of fidelity and has also been applied to safety validation of autonomous systems [13]. Multi-fidelity approaches have been shown to perform well while keeping computational requirements low, but are limited to a finite number of fidelity setting combinations [14]. In reality, however, simulators often offer dozens of parameters that determine the fidelity of a simulation. Consequently, it is infeasible to consider all possible combinations of settings.

Rather than combining the results from a limited number of simulators with expert-selected fidelity settings, our approach simultaneously learns the optimal fidelity settings (for an arbitrary simulator with many fidelity settings) while concurrently learning scenario configurations where the system under test is more likely to fail. The overall goal is to maximize the number of failures found in a given time. Our approach works with a combination of mixed continuous and discrete fidelity and scenario settings while taking the uncertainty of the outcome into account. To achieve this, we use a meta-learning framework where the task is learning the scenario parameters that lead to system failure. The optimized fidelity settings are considered as a prior that facilitate finding failures for new scenarios that have yet to be encountered during the training faster. The framework requires a high-fidelity simulator—providing the ground truth—and a learned-fidelity simulator during the

[1]Marc R. Schlichting, Anthony L. Corso, and Mykel J. Kochenderfer are with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305, USA mschl@stanford.edu

[2]Nina V. Boord and Mykel J. Kochenderfer are with the Department of Computer Science, Stanford University, Stanford, CA 94305, USA

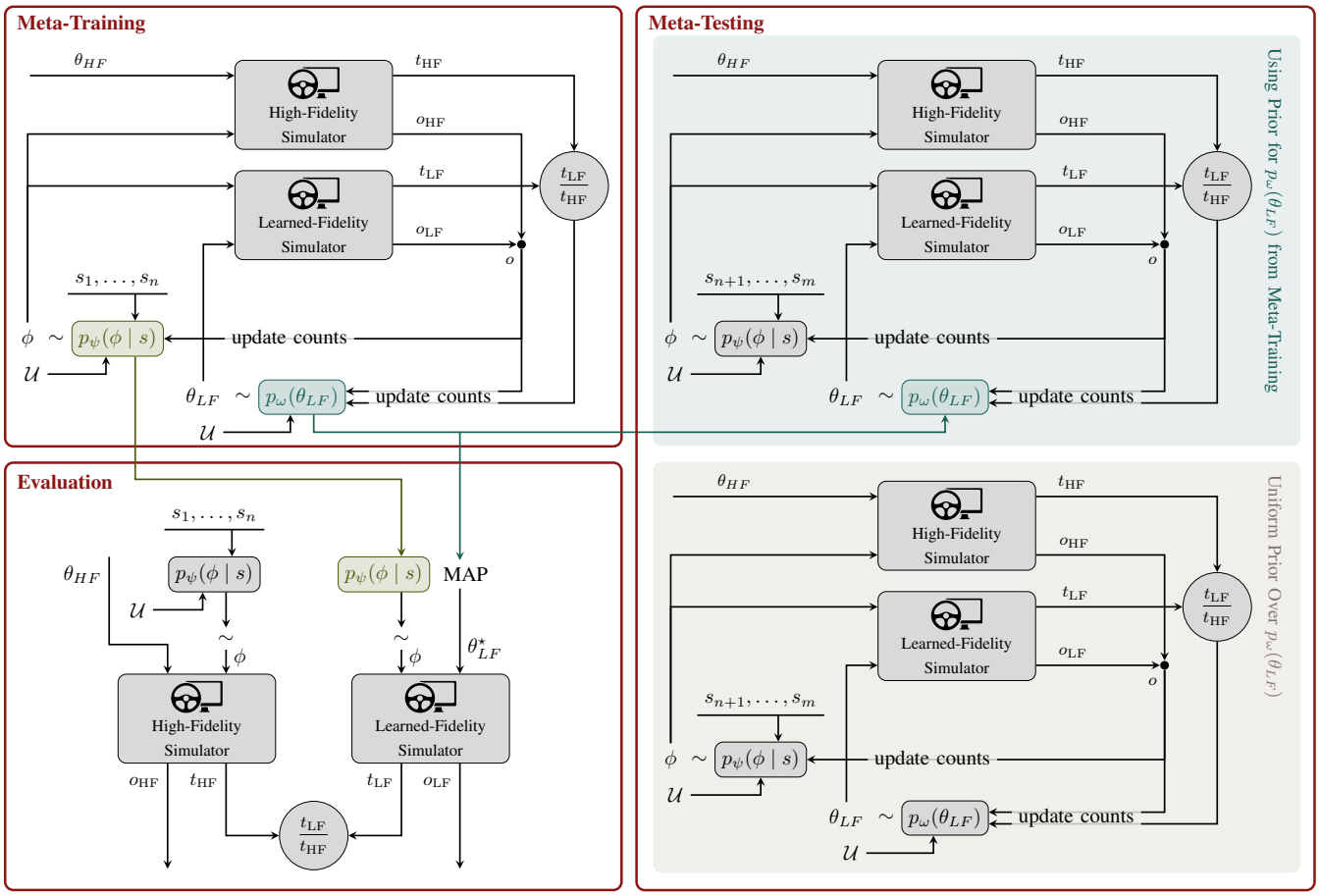

Fig. 1: Meta-learning framework for efficient safety validation.

training phase. Uncertainty is taken into account by framing the optimization problem as a multi-armed bandit problem using Bayesian model estimation and Thompson sampling.

We validate the feasibility of our approach using a cutting-edge 3D driving simulator. A total of 16 fidelity settings can be controlled, leveraging an autonomous vehicle stack comprising both camera and lidar sensors. The scenarios used for the experiments are derived from an autonomous vehicle-specific pre-crash typology [15]. Through our experiments, we demonstrate the capability of our framework to not only learn the probability distribution of failure likelihood across scenario parameters, but also the distribution for accurate and fast simulation results across the fidelity settings. Despite the computational overhead originating from the parallel use of the high-fidelity and learned fidelity simulator, we reach the break-even point—the time at which we found more failures compared to only using a high-fidelity simulator—before the end of the training phase. Through the parallel training, we reduce the average time to find a failure by a factor of 18. Furthermore, we demonstrate that the acquired distribution over the fidelity settings can be used as a warm start for learning the parameter distributions for a novel set of scenarios. Using this head start, we can accelerate the learning process up to two times over the parallel learning with uniform prior as described above.

This paper makes two key contributions. First, it presents a simulator-agnostic framework that enables us to find distributions over both scenario parameters that yield a high probability of failure and fidelity settings that yield a high probability of a fast runtime while maintaining accuracy. Second, these scenario-agnostic fidelity settings facilitate the accelerated learning of the distribution over scenario parameters for new scenarios, increasing the efficiency of validating new scenarios after training.

II. Methodology

Our framework is based on meta-learning and Bayesian approaches for multi-armed bandits. This section explains how we combine both techniques to maximize the number of failures we find with constrained compute time. Let s be an abstract scenario description and ϕ represent the scenario configuration, which is a vector that contains all values to create a runable instance of the scenario. The learned-fidelity settings—denoted by θ_{LF}—are the values of all the fidelity settings for the learned-fidelity simulator. The SAVME framework learns the scenario-specific distributions $p_\psi(\phi \mid s)$—with ψ as parameters—at the same time as the distribution over the scenario-agnostic fidelity settings $p_\omega(\theta_{\text{LF}})$, parameterized by ω. A core component of our framework is the concurrent use of a high-fidelity simulator

with fidelity settings θ_{HF} and a learned-fidelity simulator with fidelity settings θ_{LF}. To be precise, we want to solve two constrained optimization problems:

$$\begin{aligned} \underset{\psi}{\text{maximize}} \quad & \mathbb{E}_{\substack{s\sim p(s) \\ \phi\sim p_\psi(\phi|s)}} \left[\mathbb{1}\left[\mathrm{sim}(s,\phi,\theta^\star_{\mathrm{LF}},\theta_{\mathrm{HF}}) \in \mathcal{S}_{\mathrm{scenario}}\right]\right] \\ \text{subject to} \quad & \theta^\star_{\mathrm{LF}} = \underset{\theta_{\mathrm{LF}}}{\arg\max}\ p_\omega(\theta_{\mathrm{LF}}), \end{aligned} \tag{1}$$

$$\begin{aligned} \underset{\omega}{\text{maximize}} \quad & \mathbb{E}_{\substack{s\sim p(s) \\ \phi\sim p_\psi(\phi|s)}} \left[\mathbb{1}\left[\mathrm{sim}(s,\phi,\theta^\star_{\mathrm{LF}},\theta_{\mathrm{HF}}) \in \mathcal{S}_{\mathrm{fidelity}}\right]\right] \\ \text{subject to} \quad & \theta^\star_{\mathrm{LF}} = \underset{\theta_{\mathrm{LF}}}{\arg\max}\ p_\omega(\theta_{\mathrm{LF}}), \end{aligned} \tag{2}$$

where SIM is the function that simulates scenario s with parameters ϕ, using the high and learned-fidelity simulator with fidelity settings θ_{HF} and $\theta^\star_{\mathrm{LF}}$, respectively. SIM returns the outcome $o \in \{\mathrm{TP}, \mathrm{TN}, \mathrm{FP}, \mathrm{FN}\}$ and the runtimes of the high and learned-fidelity simulators, t_{HF} and t_{LF}, respectively:

$$o, t_{\mathrm{LF}}, t_{\mathrm{HF}} = \mathrm{sim}(s, \phi, \theta^\star_{\mathrm{LF}}, \theta_{\mathrm{HF}}). \tag{3}$$

For learning the scenario parameters, a success $\mathcal{S}_{\mathrm{scenario}}$ is defined as a failure of the system under test that is found using the high-fidelity simulator while for learning the fidelity settings, a success $\mathcal{S}_{\mathrm{fidelity}}$ is an outcome agreement between the high and learned-fidelity simulator in conjunction with a fast runtime. A more detailed description is given in Sections II-B and II-C and summarized in Table I. Throughout this work, we assume that $p(s)$ is uniform, meaning that all scenarios are equally likely. However, $p(s)$ can be an arbitrary distribution.

A. Meta-Learning Framework

We partition the meta-learning framework into three phases: *meta-training*, *evaluation*, and *meta-testing*. Figure 1 shows these phases of the framework.

1) Meta-Training: For each training iteration, the scenario parameters ϕ for one of the randomly selected training scenarios $s_1, \ldots, s_n$ are sampled from $p_\psi(\phi \mid s)$. In a similar fashion, we sample the learned-fidelity settings $\theta_{\mathrm{LF}} \sim p_\omega(\theta_{\mathrm{LF}})$ as well. The high-fidelity settings θ_{HF} are determined *a priori* and are never changed throughout the training, evaluation, or testing process. During the meta-training phase, we run the same scenario using both the high and learned-fidelity simulator, each simulation producing a binary outcome—o_{HF} and o_{LF}—of either *failure* or *no failure*. The respective runtimes are denoted as t_{HF} and t_{LF}. Because our objective is to identify failures in the system under test, a *failure* outcome is considered a desirable result. Given the results from the two simulators, we can enumerate all four possible combinations of outcomes o: 1) *true positive* (TP) when both simulations return *failure*, 2) *true negative* (TN) when both simulations return *no failure*, 3) *false positive* (FP) when the high-fidelity simulator returns *no failure* and the learned-fidelity simulator returns *failure*, and 4) *false negative* (FN) when the high-fidelity simulator returns *failure* and the learned-fidelity simulator returns *no failure*. These outcome cases in combination with the runtimes are used for learning the distributions $p_\psi(\phi \mid s)$ and $p_\omega(\theta_{\mathrm{LF}})$, a process which is further described in Sections II-B and II-C.

2) Evaluation: The purpose of the evaluation is to assess the performance of the learned models after meta-training. In this phase, unlike during meta-training or testing, ψ and ω are no longer updated. Furthermore, instead of sampling the fidelity settings, we greedily select them using the *maximum a posteriori* estimate (MAP) of $p_\omega(\theta_{\mathrm{LF}})$, denoted as $\theta^\star_{\mathrm{LF}}$. Note that sampling ϕ from $p_\psi(\phi \mid s)$ is still required to avoid repeatedly simulating the same scenario instance. During the evaluation phase, we can assess how well the learned $p_\psi(\phi \mid s)$ and $p_\omega(\theta_{\mathrm{LF}})$ perform compared to a baseline, such as uniform sampling from the scenario settings and simulation only through the high-fidelity simulator.

3) Meta-Testing: The purpose of the meta-testing phase is to evaluate whether the learned distribution over the fidelity settings $p_\omega(\theta_{\mathrm{LF}})$ generalizes well to a set of new scenarios $s_{n+1}, \ldots, s_m$ and therefore provides a catalyst for faster learning of $p_\psi(\phi \mid s)$ for $s \in \{s_{n+1}, \ldots, s_m\}$. Note that the meta-testing phase is still a learning process, where ψ and ω are updated, but $p_\omega(\theta_{\mathrm{LF}})$ is initialized with the posterior of $p_\omega(\theta_{\mathrm{LF}})$ from the meta-training phase while $p_\psi(\phi \mid s)$ is initialized with a uniform prior. The testing aspect focuses on whether the learned $p_\omega(\theta_{\mathrm{LF}})$ from meta-testing accelerates learning of $p_\psi(\phi \mid s)$ for new scenarios compared to initializing $p_\omega(\theta_{\mathrm{LF}})$ with a uniform distribution.

B. Fidelity Setting Learning

For learning the fidelity settings, an outcome is desirable when the high and learned-fidelity simulators produce consistent results (i.e., TP or TN). In addition, the ratio of learned-fidelity runtime over high-fidelity runtime should be less than a given compute budget $t_{\mathrm{LF}}/t_{\mathrm{HF}} \leq C_{\mathrm{budget}}$ with $C_{\mathrm{budget}} \in (0, 1]$. By using the ratio $t_{\mathrm{LF}}/t_{\mathrm{HF}}$ rather than the raw runtime, we account for the fact that the simulated time between scenarios might vary. If C_{budget} is small, we require shorter runtimes of the learned-fidelity simulation compared to the high-fidelity simulation.

We define a success for learning fidelity settings as

$$\mathcal{S}_{\mathrm{fidelity}} = \{(o, t_{\mathrm{LF}}, t_{\mathrm{HF}}) \mid o \in \{\mathrm{TP}, \mathrm{TN}\}, \frac{t_{\mathrm{LF}}}{t_{\mathrm{HF}}} \leq C_{\mathrm{budget}}\} \tag{4}$$

and a loss $\mathcal{L}_{\mathrm{fidelity}}$ consisting of $(0, t_{\mathrm{LF}}, t_{\mathrm{HF}}) \notin \mathcal{S}_{\mathrm{fidelity}}$. A summary is provided in Table I. If C_{budget} is chosen too small, we might encounter difficulty learning $p_\omega(\theta_{\mathrm{LF}})$ because the multi-armed bandit formulation requires successful trials as defined in Eq. (4). In most cases, there exists a lower bound $\underline{\mathrm{C}}_{\mathrm{budget}}$ where for any $C_{\mathrm{budget}} < \underline{\mathrm{C}}_{\mathrm{bugdet}}$ no successful trial can be encountered.

Under the assumption that each fidelity setting independently affects the behavior of a simulation, we model the problem as a multi-armed bandit problem [16]. Due to the independence assumptions between the different fidelity settings, we can solve $|\theta_{\mathrm{LF}}|$ independent multi-armed bandit problems, where the number of arms for each problem i represents the number of values a fidelity setting $\theta_{LF,i}$

TABLE I: Success $\mathcal{S}$ and loss $\mathcal{L}$ for learning $p_\psi(\phi \mid s)$ and $p_\omega(\theta_{\mathrm{LF}})$

	Task	
	Fidelity Settings	Scenario Parameters
$\mathcal{S}$	$(o \in \{\mathrm{TP}, \mathrm{TN}\}) \wedge \left(t_{\mathrm{LF}}/t_{\mathrm{HF}} \leq C_{\mathrm{budget}}\right)$	$o \in \{\mathrm{TP}, \mathrm{FN}\}$
$\mathcal{L}$	$(o \in \{\mathrm{FP}, \mathrm{FN}\}) \vee \left(t_{\mathrm{LF}}/t_{\mathrm{HF}} > C_{\mathrm{budget}}\right)$	$o \in \{\mathrm{TN}, \mathrm{FP}\}$

can take on. Although many fidelity settings are binary or categorical and well-suited for multi-armed bandit problems, other settings, such as distances, are continuous. By discretizing continuous fidelity settings, however, we can still use them with the multi-armed bandit framework.

The belief over the success for each possible fidelity setting value is represented by a beta distribution where $\alpha = n_{\mathrm{prior},\mathcal{S}} + n_{\mathcal{S}}$ and $\beta = n_{\mathrm{prior},\mathcal{L}} + n_{\mathcal{L}}$ with $n_{\mathcal{S}}$ and $n_{\mathcal{L}}$ being the counts of successes and losses as defined in Table I, respectively. Prior knowledge about the distribution can be incorporated by adjusting $n_{\mathrm{prior},\mathcal{S}}$ and $n_{\mathrm{prior}_{\mathcal{L}}}$, whereas a uniform prior is equal to $n_{\mathrm{prior},\mathcal{S}} = n_{\mathrm{prior},\mathcal{L}} = 1$.

To extract the optimal fidelity settings, the distribution over θ_{LF} needs to be learned as accurately as possible. We use Thompson sampling [17], [18] to balance exploration and exploitation during training. For discrete fidelity settings, we can use Thompson sampling as described in the literature. For continuous fidelity settings, we use a stratified sampling scheme where the bin is sampled through Thompson sampling and the actual fidelity setting value is sampled within the bin according to either a uniform distribution (if uniform intervals were chosen) or a log-uniform distribution (if logarithmic intervals were chosen).

During the evaluation phase, there is no further need for exploration, and the *optimal* fidelity settings can be selected using the greedy MAP estimate for each fidelity setting:

$$\theta^{\star}_{LF,i} = \arg\max_j \frac{\alpha_j - 1}{\alpha_j + \beta_j - 2}, \tag{5}$$

where i is indexing the fidelity setting and j is indexing the possible values for each fidelity setting. For continuous variables, we determine the expected value based on either a uniform or log-uniform distribution.

C. Learning Failure Scenarios

The method for learning failure scenarios—scenarios that are likely to lead to a failure of the system under test—is similar to the fidelity-learning framework described in Section II-B with three differences:

1) A success $\mathcal{S}_{\mathrm{scenario}}$ is only dependent on the outcome and is defined when the outome o is either TP or FN. Consequently a loss $\mathcal{L}_{\mathrm{scenario}}$ is defined when the outcome o is either TN or FP.
2) As each scenario type can have different parameters, a distribution $p_\psi(\phi)$ must be learned for each scenario s which can be written as a conditional distribution $p_\psi(\phi \mid s)$.
3) Instead of using the MAP estimate of the learned distribution during evaluation, we sample the scenarios from the learned distribution to prevent running the same scenario instance repeatedly.

III. Experiments

We demonstrate the feasibility of our approach by using a state-of-the-art 3D autonomous driving simulator by Applied Intuition,[1] a widely adopted platform in the autonomous driving industry. All experiments are run on a machine with an Intel Core i9-9900KF CPU, 64GB RAM, and a NVIDIA RTX 2080 Ti GPU. SAVME is simulator-agnostic and we provide starter code with a generic simulator and instructions.[2]

A. System under Test

We use a stack that consists of a localization sensor, a camera sensor, and a lidar sensor to demonstrate that our approach is capable of handling complex systems with many fidelity settings. No motion planning is required because the desired lateral path is specified in the scenarios. For lateral path tracking, we use a Stanley controller [19], and for longitudinal control we use a PI controller that keeps the vehicle at a desired speed if no obstacle is detected. In the case of a predicted collision, a constant braking force is applied. The collision detection prediction is a two-stage process which begins with analyzing the camera image using a pre-trained YOLOv5s object detection network [20] trained on the widely-used COCO object detection dataset [21].

If an object of the category *vehicle* or *person* is detected, the lidar signal is filtered according to the discovered bounding box and the weighted centroid of the filtered lidar pointcloud is taken as measurement for the obstacle's position. The weights are proportional to the intensity of the point as reported by the lidar sensor. We predict the closest distance between the ego vehicle and the obstacle based on a first-order point-mass model which is fitted to the current and previous obstacle position. If the predicted minimal distance is less than a threshold of one car length, the brake event is triggered. We choose this safety buffer to account for noisy measurements and the inaccuracies of the first-order model. A more detailed description can be found in the repository.

B. Scenarios

Instead of the NHTSA pre-crash typology [22] which is based on crashes between human-operated cars, we use a pre-crash typology from crash reports of incidents involving autonomous vehicles [15]. A total of 10 scenario types where the autonomous vehicle plays an active role in the accident are used for our assessment. No scenarios in which the autonomous vehicle is passive are included, such as instances where it is rear-ended. Out of the 10 scenarios, we use 8 scenarios for the meta-training phase and 2 scenarios for the meta-testing phase. The scenarios are depicted in Fig. 2 where scenarios 1 through 8 are the scenarios that are used for the meta-training while scenarios 9 and 10 are used for meta-testing.

[1] https://www.appliedintuition.com
[2] https://github.com/sisl/SAVME.git

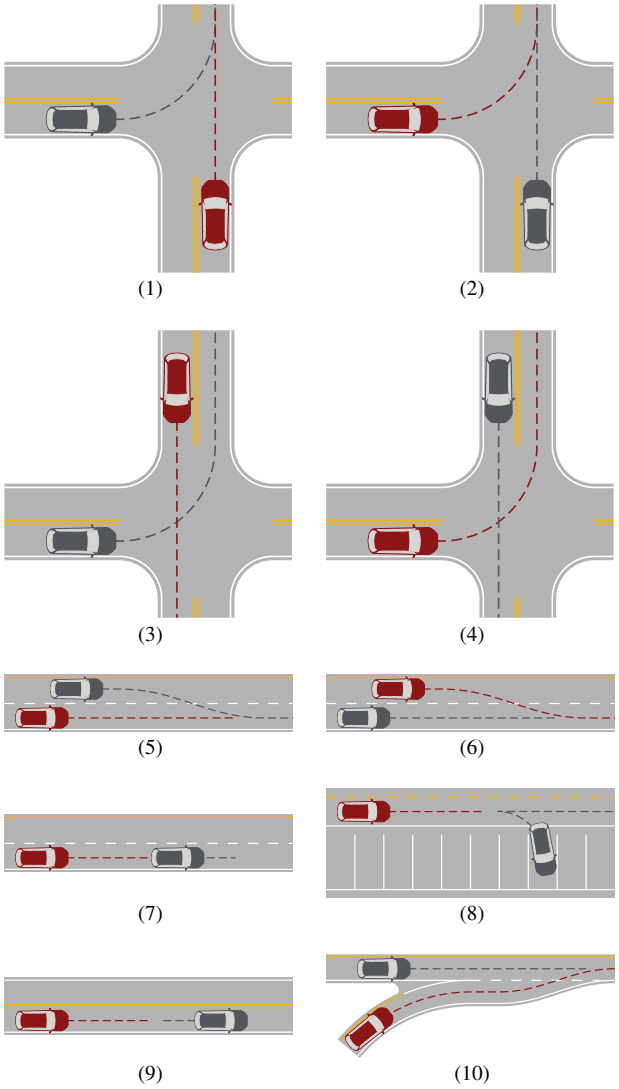

Fig. 2: Pre-crash scenarios. Scenarios 1 through 8 are used for meta-training, while scenarios 9 and 10 are used for meta-testing. The red vehicle represents the ego vehicle while the gray vehicle is the obstacle.

TABLE II: Fidelity settings with θ_{HF} highlighted

Sensor	Fidelity Setting	Type	Values/Range
-	simulation rate	discrete	{2, 4, 6, 8, **10**} Hz
camera	bloom level	discrete	{**high**, low}
camera	disable bloom	discrete	{true, **false**}
camera	disable lighting	discrete	{true, **false**}
camera	disable shadows	discrete	{true, **false**}
camera	disable lens model	discrete	{true, **false**}
camera	disable depth of field	discrete	{true, **false**}
camera	disable shot noise	discrete	{true, **false**}
camera	view distance	continuous	[10, **5000**] m
camera	near clipping distance	continuous	[**0.2**, 20] m
lidar	disable shot noise	discrete	{true, **false**}
lidar	disable ambient effects	discrete	{true, **false**}
lidar	disable translucency	discrete	{true, **false**}
lidar	subsample count	discrete	{1, 2, 3, 4, **5**}
lidar	raytracing bounces	discrete	{0, 1, 2, 3, **4**}
lidar	near clipping distance	continuous	[**0.2**, 20] m

C. Fidelity Settings.

To demonstrate the feasibility of our framework in a high-dimensional, mixed categorical and continuous fidelity space, we use the 16 fidelity settings in Table II alongside the definition for θ_{HF}.

D. Baseline and Experiment Goals

As a baseline, we sample the scenario settings ϕ from a uniform distribution and evaluate those using the high-fidelity settings θ_{HF} that correspond to the simulator's recommended settings. We can formulate two experimental goals based on the results from the meta-training and meta-testing phases:

1) The evaluation of the meta-training phase reveals how well the proposed framework can detect scenarios that lead to failures, i.e., learning $p_\psi(\phi \mid s)$, while also making each simulation run faster by adjusting the the fidelity settings, i.e., learning $p_\omega(\theta_{\text{LF}})$. At the end of the meta-training phase, it is possible to calculate the speedup factor, which denotes the increase in the number of detected failures within the same time span as compared to the baseline. Of further interest is the break-even point, or time at which we detect the same number of failures using our dual high and learned-fidelity simulators than we did with the baseline.
2) While $p_\psi(\phi \mid s)$ is conditional on the scenario, $p_\omega(\theta_{\text{LF}})$ is scenario-agnostic and therefore used across all scenarios during the meta-training phase. During the meta-testing phase, we evaluate whether using the learned $p_\omega(\theta_{\text{LF}})$ from meta-training as a prior helps to speed up learning $p_\psi(\phi \mid s)$ on unseen scenarios.

Our findings for both goals are presented in Section IV.

IV. Results

A. Meta-Training

By isolating the results of the meta-training phase, we can understand the suitability of the multi-armed bandit approach for simultaneously learning $p_\psi(\phi \mid s)$ and $p_\omega(\theta_{\text{LF}})$. We train both $p_\psi(\phi \mid s)$ and $p_\omega(\theta_{\text{LF}})$ using scenarios 1 through 8 as shown in Fig. 2 for 500 iterations where the probability $p(s)$ is uniform. We evaluate the learned $p_\psi(\phi \mid s)$ and $p_\omega(\theta_{\text{LF}})$ using 100 evaluations with $\phi \sim p_\psi(\phi \mid s)$ and $\theta^\star_{\text{LF}} = \arg\max_{\theta_{\text{LF}}} p_\omega(\theta_{\text{LF}})$. For each $C_{\text{budget}} \in \{0.2, 0.3, 0.4\}$, we calculate the TP-rate (how many failures we would've found only using the learned-fidelity simulator), the mean relative runtime of the learned-fidelity simulator when compared against the high-fidelity simulator, and the relative speedup from the baseline. All results are shown in Table III. We first note that for all C_{budget}, the mean learned-fidelity cost is remarkably similar given the different bounds. Second, we note the significant performance drop for $C_{\text{budget}} = 0.2$ which indicates that 0.2 approaches $\underline{C}_{\text{budget}}$. Finally, an almost 18 times speedup relative baseline is achieved with $C_{\text{budget}} = 0.3$. In other words, at evaluation, using the learned-fidelity simulator, we found almost 18 times as many failures within the same runtime.

Concurrently executing the high-fidelity and learned-fidelity simulators during meta-training incurs significant computational cost. Thus, it is important to determine the runtime threshold at which our approach exhibits a higher incidence of failures compared to the baseline, also called the break-even point. The break-even point can either occur during or after training. Figure 3 shows the failures over the runtime for the baseline and for the meta-training phase with $C_{\text{budget}} = \{0.2, 0.3, 0.4\}$, each executed for 500 iterations. We conclude that the break-even point occurs before the end of the meta-training phase between $50\,\text{min}$ and $80\,\text{min}$ for all C_{budget}. In other words, despite the concurrent usage of the high and learned-fidelity simulators, we find more failures during the meta-training phase than the baseline.

Table IV shows the learned fidelity settings. While some settings differ based on C_{budget}, most settings remain the same. Finally, Fig. 4 shows an example of a camera image contrasting the difference between the high and learned-fidelity settings.

B. Meta-Testing

For meta-testing, we use scenarios 9 and 10 from Fig. 2 to evaluate the effect of using the posterior distribution $p_\omega(\theta_{\text{LF}})$ from the meta-training phase as prior for the meta-testing phase. We compare against the case of using a uniform prior for $p_\omega(\theta_{\text{LF}})$ as well as the previously introduced baseline. The meta-testing phase has a duration of 200 iterations where we record the runtime and the found failures. The results are depicted in Fig. 5. Using the learned prior for $p_\omega(\theta_{\text{LF}})$ results in more found failures for all compute budgets $C_{\text{budget}} = \{0.2, 0.3, 0.4\}$ during the meta-testing phase when compared to using a uniform prior. We thus arrive at the conclusion that utilizing the posterior $p_\omega(\theta_{\text{LF}})$ obtained during the meta-training phase as the prior for $p_\omega(\theta_{\text{LF}})$ while learning $p_\psi(\phi \mid s)$ for new scenarios $s_{n+1}, \ldots, s_m$ exhibits a beneficial impact on the rate of learning. For our experiments, we observe an increase of 50 to 100%. We also included the baseline that was introduced in Section III-D to demonstrate the two levels from this study: The comparison of the uniform prior with the baseline is the setup during meta-training while the comparison between a uniform prior and the learned prior is the setup for meta-testing. Figure 5 can be seen as the summary of this entire study. We demonstrate that even with a uniform prior on $p_\omega(\theta_{\text{LF}})$, the SAVME framework performs better than the baseline, but using the learned prior on $p_\omega(\theta_{\text{LF}})$, leads to an even more increased speedup.

C. Limitations

We acknowledge two primary limitations inherent in our present approach. First, because we are using a multi-armed bandit framework, we are restricted to discrete or discretized scenario parameters and fidelity settings. This limitation could be overcome by using a more general Dirichlet process formulation. Second, assuming independence within and between the scenario parameters and fidelity settings might be an oversimplification that can necessitate the use of

TABLE III: Results from evaluation after meta-training

C_{budget}	TP-Rate	Mean LF Cost	Speedup
baseline	0.17	1.00	1.00
0.4	0.41	0.26	9.32
0.3	**0.74**	0.25	**17.99**
0.2	0.29	**0.22**	7.86

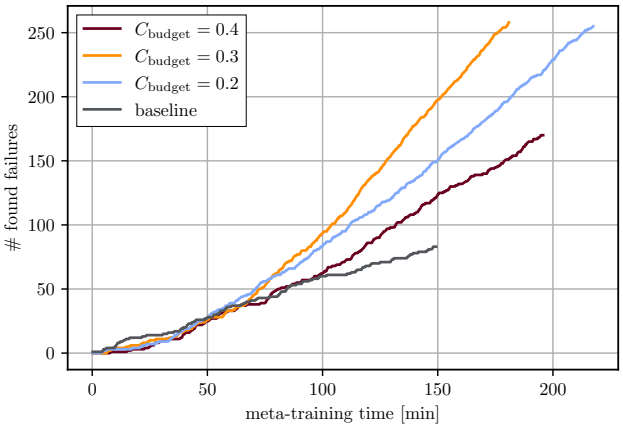

Fig. 3: Found failures over runtime during the meta-training phase. Each experiment is 500 iterations long. For all C_{budget}, the break-even point is reached before the end of the training.

TABLE IV: Learned fidelity settings

Sensor	Fidelity Setting	$C_{b.} = 0.4$	$C_{b.} = 0.3$	$C_{b.} = 0.2$
-	simulation rate	2 Hz	2 Hz	2 Hz
camera	bloom level	low	high	low
camera	disable bloom	true	true	false
camera	disable lighting	true	false	false
camera	disable shadows	true	true	true
camera	disable lens model	false	false	true
camera	disable depth of field	true	false	false
camera	disable shot noise	true	true	false
camera	view distance	65.70 m	285.97 m	65.70 m
camera	near clipping distance	3.67 m	7.22 m	0.58 m
lidar	disable shot noise	false	true	false
lidar	disable ambient effects	false	false	false
lidar	disable translucency	true	false	false
lidar	subsample count	3	2	1
lidar	raytracing bounces	0	0	1
lidar	near clipping distance	1.68 m	1.68 m	13.57 m

(a) Camera image with θ_{HF} (b) Camera image with $\theta^\star_{\text{LF}}$

Fig. 4: Comparison of images from the camera sensor between θ_{HF} and $\theta^\star_{\text{LF}}$ for $C_{\text{budget}} = 0.3$. The differences can be seen especially well in the maximum view distance and reflections.

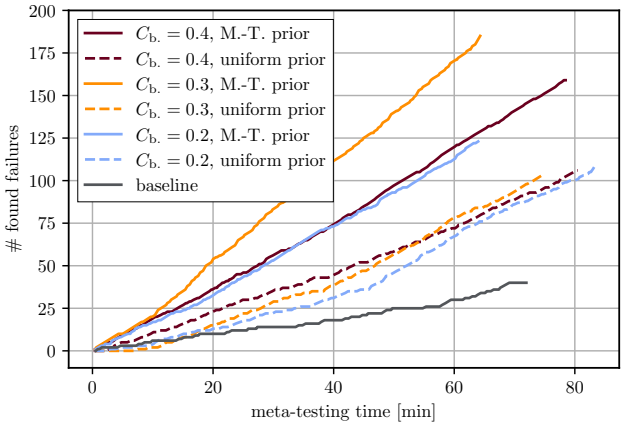

Fig. 5: Found failures over runtime during meta-testing. Each experiment is 200 iterations long. Using the posterior of $p_\omega(\theta_{\text{LF}})$ from meta-training as prior during meta-testing resulted in faster learning of $p_\psi(\phi \mid s)$ for all C_{budget}.

more advanced techniques involving probabilistic graphical models. While it is true that many applications may not be significantly impacted by these limitations, we believe they serve as a valuable foundation for expanding the potential applications of the SAVME framework in the future.

V. Conclusion

This paper presents an efficient, falsification-based validation approach for autonomous systems using meta-learning in conjunction with a Bayesian multi-armed bandit formulation. Our framework is unique as we are approaching efficient safety validation from the falsification perspective as well as from the efficient simulation perspective. In addition, with our approach, the learned scenario-agnostic fidelity settings can be used to accelerate the falsification process for new scenarios, which represents the meta aspect of our method. The SAVME framework's source code is open-source and requires minimal effort to apply to other problems.

In our experiments we use a state-of-the-art 3D driving simulator and a driving stack with camera and lidar sensors. The scenarios come from an AV-specific pre-crash typology while the simulation setup provides 16 fidelity settings. During the evaluation following meta-training, our framework demonstrates on average an almost 18-fold reduction in the time required to detect a failure. Furthermore, we find that despite the concurrent use of two simulators, more failures can be discovered even during the training phase when compared to the baseline that only uses one simulator. During the meta-learning phase, we observe that using the fidelity settings obtained during meta-training as a prior, increases the falsification process up to a factor of two.

Acknowledgements

The authors would like to express their sincere gratitude to Allstate for their generous funding support through the Stanford Center for AI Safety as well as to Applied Intuition for the software access and technical support.

References

[1] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 182–193, 2016.

[2] S. Sovani, "Simulation accelerates development of autonomous driving," *ATZ Worldwide*, vol. 119, no. 9, pp. 24–29, 2017.

[3] A. Corso, R. Moss, M. Koren, R. Lee, and M. Kochenderfer, "A survey of algorithms for black-box safety validation of cyber-physical systems," *Journal of Artificial Intelligence Research*, vol. 72, pp. 377–428, 2021.

[4] F. Rosique, P. J. Navarro, C. Fernández, and A. Padilla, "A systematic review of perception system and simulators for autonomous vehicles research," *Sensors*, vol. 19, no. 3, p. 648, 2019.

[5] M. J. Kochenderfer and T. A. Wheeler, *Algorithms for Optimization*. MIT Press, 2019.

[6] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems," in *International Conference on Computer Aided Verification*, 2010, pp. 167–170.

[7] Y. S. R. Annapureddy and G. E. Fainekos, "Ant colonies for temporal logic falsification of hybrid systems," in *Annual Conference on IEEE Industrial Electronics Society*, 2010, pp. 91–96.

[8] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal Control*. John Wiley & Sons, 2012.

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

[10] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on Robot Learning*, 2017, pp. 1–16.

[11] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics: Results of the 11th International Conference*, 2018, pp. 621–635.

[12] J. Xu, S. Zhang, E. Huang, C.-H. Chen, L. H. Lee, and N. Celik, "Efficient multi-fidelity simulation optimization," in *Winter Simulation Conference*, 2014, pp. 3940–3951.

[13] J. J. Beard and A. Baheri, "Safety verification of autonomous systems: A multi-fidelity reinforcement learning approach," *arXiv preprint arXiv:2203.03451*, 2022.

[14] R. Pellegrini, J. Wackers, R. Broglia, A. Serani, M. Visonneau, and M. Diez, "A multi-fidelity active learning method for global design optimization problems with noisy evaluations," *Engineering with Computers*, pp. 1–24, 2022.

[15] Q. Liu, X. Wang, X. Wu, Y. Glaser, and L. He, "Crash comparison of autonomous and conventional vehicles using pre-crash scenario typology," *Accident Analysis & Prevention*, vol. 159, p. 106281, 2021.

[16] H. Robbins, "Some aspects of the sequential design of experiments," *Bulletin of the American Mathematical Society*, vol. 55, pp. 527–535, 1952.

[17] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3-4, pp. 285–294, 1933.

[18] S. Agrawal and N. Goyal, "Analysis of thompson sampling for the multi-armed bandit problem," in *Conference on Learning Theory*, 2012, pp. 39.1–39.26.

[19] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun, "Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing," in *American Control Conference*, 2007, pp. 2296–2301.

[20] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[21] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision*, 2014, pp. 740–755.

[22] W. G. Najm, J. D. Smith, M. Yanagisawa *et al.*, "Pre-crash scenario typology for crash avoidance research," United States. National Highway Traffic Safety Administration, Tech. Rep., 2007.