# Multi-Fidelity Bayesian Optimization for Simulation Based Autonomous Driving Systems Testing

Olek Osikowicz*, Phil McMinn*, Wei Xing†, Donghwan Shin*
*School of Computer Science, †School of Mathematical and Physical Sciences, University of Sheffield, Sheffield, UK
{amosikowicz1, p.mcminn, w.xing, d.shin}@sheffield.ac.uk

***Abstract*—Simulation-based testing has become a powerful method for uncovering critical failures in Autonomous Driving Systems (ADS). However, the high computational cost of executing realistic driving simulations, which often takes minutes if not hours, at full fidelity significantly limits the scalability of this approach. In this work, we introduce MFBO-Drive, a novel approach that formulates critical scenario generation as a multi-fidelity optimization problem. By adaptively selecting both candidate test scenarios and their corresponding simulation fidelity levels, MFBO-Drive enhances the efficiency of ADS testing without compromising effectiveness. MFBO-Drive is instantiated using the well-established Multi-Fidelity Bayesian Optimization (MFBO) method and incorporates a fidelity exploration control mechanism to balance simulation cost with predictive reliability. We evaluate the approach on over 100,000 driving scenarios in the MetaDrive simulator, comparing it against strong baselines including single-fidelity Bayesian optimization and random search. Experimental results show that MFBO-Drive significantly enhances cost-effectiveness, achieving a 16.8% improvement over state-of-the-art Bayesian optimization, while maintaining comparable test quality. These results highlight its promise for budget-constrained ADS testing in simulation environments.**

***Index Terms*—Autonomous Driving Systems, Search-based Software Testing, Multi-fidelity Bayesian Optimization.**

## I. Introduction

Simulation-based testing for Autonomous Driving Systems (ADS) has been actively studied [1, 2] to ensure the safety and reliability of ADS while reducing the costs and risks involved in real-world testing. For example, a challenging driving scenario involving interactions with lots of moving vehicles and pedestrians can be simulated as a test scenario in a driving simulator, such as MetaDrive [3] or CARLA [4], to evaluate the ADS under test.

One can generate challenging driving scenarios automatically by tweaking various scenario entities (e.g., road shapes, the trajectories of other vehicles) with the goal of maximizing the degree of safety violations (e.g., the distance from the center of the lane) using optimization algorithms (e.g., Genetic Algorithms and Bayesian Optimization). It has been actively researched [5, 6], and naturally, they have relied on high-fidelity driving simulators to mimic the real world as closely as possible. However, high-fidelity simulations are often computationally expensive, leading to significant time and resource costs, especially when evaluating a large number of scenarios. This raises an essential question: *Is the highest simulation fidelity really necessary for evaluating safety violations across all candidate scenarios?*

For example, decreasing a simulation's frame rate (i.e., the frequency of updating the simulated world) from 60 Frames Per Second (FPS) to 10 FPS could dramatically lower simulation costs (time and resource) without significantly affecting the assessment results for some test scenarios. Nevertheless, to the best of our knowledge, none of the existing simulation-based ADS testing studies have specifically leveraged *lower-fidelity simulations* to minimize simulation costs while maintaining the same level of test effectiveness.

In this paper, we answer the above question by presenting a novel approach that dynamically varies simulation fidelity to generate critical scenarios more efficiently. The key assumption behind the approach is that not all test scenarios would be significantly affected by lowering simulation fidelity; in other words, there would be some scenarios where there is little or no difference between high- and low-fidelity scenario assessments. If we could predict which scenario evaluations are not to be affected negatively by the change in simulation fidelity, we could effectively use lower-fidelity simulations for as many test scenarios as possible, minimizing the simulation costs without sacrificing the test accuracy. To address this, we propose *MFBO-Drive*, which dynamically varies simulation fidelity to uncover critical scenarios. We adapt Multi-Fidelity Bayesian Optimization (MFBO) [7, 8] by incorporating a domain-specific *fidelity exploitation rate* ($\epsilon$). This balances low-fidelity exploitation to address the specific cost-accuracy trade-offs of ADS testing. Our evaluation on MetaDrive demonstrates that this approach increases the cost-effectiveness of critical scenario generation by up to 16.84%.

The main contributions of this paper are as follows.

(1) We propose MFBO-Drive, a novel ADS testing framework that efficiently generates challenging driving scenarios while minimizing cost of driving simulations (Section III).
(2) We provide empirical results demonstrating the effectiveness and efficiency of our approach (Section IV). The results highlight its significant potential to accelerate simulation-based ADS testing.
(3) We present a publicly available replication package, that includes a large simulation dataset (over 300,000 scenario simulations requiring more than 300 compute hours) along with the scripts allowing to replicate our experiments (Section IV-F3).

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No. RS-2025-02218761, 50%) and by the Engineering and Physical Sciences Research Council (EPSRC) [EP/Y014219/1, EP/X024539/1].

## II. Background

### A. Scenario-based ADS Testing in Driving Simulators

Scenario-based ADS testing in simulators involves three main steps [9]: *generation*, *execution*, and *evaluation*. First, a candidate test scenario is generated, defining a sequence of events (e.g., trajectories of other vehicles) happening in a specific environment (e.g., a T-junction with no traffic lights on a foggy day). Next, the scenario is executed by a simulator like MetaDrive [3] or CARLA [4], where the ADS-equipped "ego vehicle" is placed in a virtual world created by the simulator, interacting with other static and dynamic entities in the scenario. Finally, the driving log data produced during the scenario execution is evaluated for safety violations (e.g., collisions with other vehicles).

In the scenario execution step, the frequency at which the virtual environment is updated (and the ego vehicle's actions are applied) is controlled by one of the *fidelity* parameters, typically denoted as frames-per-second (FPS)[1]. Higher fidelity yields more realistic results but increases computational cost, while lower fidelity is faster but less accurate.

Finding critical scenarios that reveal safety violations is challenging due to the sheer size and complexity of the scenario space. *Search-based testing* frames this challenge as a black-box optimization problem [6, 10]. Specifically, let $X$ denote the driving scenario domain, representing the space of all possible driving scenarios. We consider a black-box objective function $f : X \to \mathbb{R}$, commonly referred to as a fitness function, which quantifies the degree of safety violations of a scenario $x \in X$. Although the inverse of $f$ is not analytically available, we can evaluate $f(x)$, i.e., the fitness score of a given scenario $x \in X$, by executing the ADS within a driving simulator. Therefore, the problem of generating critical scenarios can be transformed into a black-box optimization problem: identifying $x^* = \operatorname{argmax}_{x \in X} f(x)$ where $x^*$ denotes the globally optimal solution, i.e., a scenario with the maximum degree of safety violations.

A naive approach to approximate the optimal solution is Random Search (RS), which repeatedly samples scenarios uniformly at random from $X$ and evaluates them using $f$ until the search budget is exhausted. More sophisticated methods like Bayesian Optimization are typically more efficient.

### B. Bayesian Optimization

Bayesian Optimization (BO) is a black-box optimization technique suitable for a computationally expensive objective function that takes a long time (minutes or hours) to evaluate [11]. Instead of sampling the solution space randomly, BO constructs a computationally inexpensive *surrogate model* based on previously evaluated samples to approximate the objective function and guide the selection of future candidates, while considering prediction uncertainty, using an *acquisition function*. It iteratively refines the surrogate model as more samples are evaluated, enabling efficient exploration of the solution space. Below we detail a surrogate model and an acquisition function.

*1) Surrogate Models:* A surrogate model $\mu$ serves as an efficient approximation of the true objective function $f$. Traditionally, Gaussian Process (GP) regression has been the most widely used surrogate model in BO [12, 13], because it provides both the predictive mean $\mu(x)$ and uncertainty $\sigma(x)$. However, as noted by Frazier [11], GPs are only effective when input dimension is below 20, beyond which prediction efficiency and accuracy degrade. Alternatives such as Random Forest and Gradient Boosting are evaluated in Section IV.

*2) Acquisition Functions:* An acquisition function $\alpha : X \to \mathbb{R}$ determines which candidate solution to evaluate next on the expensive objective function $f$. For a candidate solution $x \in X$, it considers not only the surrogate model's prediction $\mu(x)$ but also its prediction uncertainty $\sigma(x)$ to balance exploration (high uncertainty) and exploitation (high predicted score). Specifically, for a given $\alpha$, the next candidate $x'$ is selected as $x' = \operatorname{argmax}_{x \in X} \alpha(x)$. Although evaluating $\alpha(x)$ is computationally inexpensive thanks to the surrogate model, globally optimizing $\alpha$ over $X$ is often infeasible, especially when the search space $X$ is continuous or high-dimensional. A common practical approach is to use Monte Carlo sampling: randomly sample $m$ candidate solutions from $X$, evaluate the acquisition function on each, and select the most promising one based on these evaluations.

One of the most widely used acquisition functions is the Upper Confidence Bound (UCB), defined as $\alpha_{\mathrm{UCB}}(x) = \mu(x) + \sigma(x)$. It assigns equal weight to exploitation and exploration, favouring data points with high fitness estimates or high uncertainty. For example, early in the search, when uncertainty is high due to limited data, UCB promotes exploration of under-sampled regions. As more evaluations are gathered and the surrogate becomes more confident, the influence of $\sigma(x)$ diminishes, and UCB shifts toward exploitation by prioritizing candidates with higher predicted fitness.

### C. Multi-fidelity Bayesian Optimization

On top of standard BO, Multi-fidelity Bayesian Optimization (MFBO) additionally takes advantage of a practical observation: the objective function $f$ often depends on a *fidelity parameter* (e.g., FPS in simulation-based ADS testing, as discussed in Section II-A) which controls a trade-off between evaluation cost and result accuracy [11, 14].

More specifically, let $A$ denote the fidelity domain, representing all possible levels of evaluation fidelity. The specific fidelity domain $A$ depends on the simulator itself and is independent of the input domain $X$ (i.e., driving scenarios). The black-box objective function $f$ now accepts an additional parameter $a \in A$, which controls the trade-off between evaluation accuracy and computational cost. In other words, for the same candidate $x \in X$, the value of $f(x, a)$ may differ depending on the chosen fidelity level $a$. The cost of evaluating $f(x, a)$ is described by a cost function $c(a)$, which is typically monotonic in $a$. Therefore, the problem is to solve

[1]Note that FPS refers to simulated time. For example, 10 FPS means that the sensor data is rendered 10 times per simulation-second. However, on a given machine, longer simulation time means longer real-world execution time.

$\text{argmax}_{x \in X} f(x, a^*)$, where $a^*$ denotes the "oracle" setting, i.e., the highest fidelity level that enables the most accurate and expensive version of $f$.

However, instead of evaluating only at the highest fidelity, MFBO strategically chooses inputs $x_i \in X$ and the corresponding fidelity levels $a_i \in A$. It then observes outputs $y_i = f(x_i, a_i)$ while aiming to minimize the total evaluation cost $\sum_{i=1}^{N} c(a_i)$ or remain within a given budget $B$. To achieve this, MFBO relies on a Multi-Fidelity (MF) surrogate model and a corresponding MF acquisition function. A common approach [8, 15] is to construct one surrogate model $\mu(x, a)$ that jointly approximates $f$ over both input $x$ and fidelity level $a$. Once $\mu(x, a)$ is trained, predictions at the highest fidelity can be obtained directly via $\mu(x, a^*)$. This enables the selection of the next candidate using any standard (single-fidelity) acquisition function, evaluated at the oracle fidelity: $x' = \text{argmax}_{x \in X} \alpha(x, a^*)$. The corresponding fidelity level $a'$ at which to evaluate $x'$ can then be selected using a fidelity query function $\gamma$, such as $a' = \max_{a \in A} \gamma(x', a)$, where $\gamma(x', a)$ quantifies the gain of evaluating $x'$ at $a$ [16].

## III. MFBO-Drive Approach

A major bottleneck in scenario-based ADS testing is the high computational cost of executing driving scenarios in realistic, high-fidelity simulators. However, we observe that (1) modern simulators often expose dedicated parameters that control simulation fidelity, and (2) these parameters directly impact both the computational cost of simulation and the behavior of the ADS under test. For example, reducing the simulation *frame rate* (FPS) can substantially lower computational cost, but it may also alter how the ADS perceives and reacts to the environment.

These key observations motivate a shift from traditional single-fidelity simulation-based testing to a multi-fidelity testing paradigm. This reformulation enables more efficient use of computational resources by balancing the *information gain* from evaluating a candidate scenario and the *cost* of running the simulator at a given fidelity. To realize this strategy, we present MFBO-Drive, a novel simulation-based ADS testing approach that builds on the well-established MFBO method, introduced in Section II-C. Our approach targets the ADS testing domain with simulators that expose a finite, ordered set of fidelity parameters, yet remains applicable to other domains.

### A. Fidelity Exploration Control

Although applying MFBO to ADS testing appears straightforward once a simulator's fidelity parameter is identified, our preliminary study revealed a key practical limitation. Standard MFBO tends to *over-exploit* low-fidelity evaluations in scenario-based ADS testing, particularly in the early stages of optimization when the surrogate model lacks sufficient high-fidelity data. This imbalance can lead to the surrogate model prematurely converging on misleading solutions, resulting in suboptimal scenario selection.

To mitigate this issue, MFBO-Drive adopts a strategy inspired by the $\varepsilon$-greedy algorithm from reinforcement learning. Specifically, we introduce a parameter $\varepsilon$, which defines the probability of forcing a high-fidelity evaluation, regardless of the outcome of the fidelity query function $\gamma$ discussed in Section II-C. This mechanism ensures minimum exploration of the oracle fidelity level and prevents the approach from becoming overly reliant on noisy or imprecise low-fidelity predictions.

### B. MFBO-Drive: Algorithms

Algorithm 1 presents the pseudocode for MFBO-Drive. It takes as input the scenario domain $X$, the fidelity domain $A$, a total search budget $b_s$, and an initialization budget $b_i$. The algorithm returns the most critical driving scenario $\hat{x}^* \in X$, selected based on evaluations collected until the budget $b_s$ is exhausted. Internally, it builds and updates surrogate models throughout the optimization process. The models are initially trained on data from the first $b_i$ units of the budget.

**Algorithm 1:** MFBO-Drive

**Input :** Scenario Domain $X$, Fidelity Domain $A$, Search Budget $b_s$, Initialization Budget $b_i$
**Output:** Most Critical Driving Scenario $\hat{x}^*$

```
1  Running Cost c ← 0
2  Set of Evaluated Scenarios D ← ∅
3  while c < b_s do
4      if c < b_i then
5          Candidate Scenario x' ← random(X)
6          Candidate Fidelity Level a' ← random(A)
7      else
8          (x', a') ← getNextScenarioFidelity(D, X, A)
9      y' ← f(x', a')
10     D ← D ∪ {(x', a', y')}
11     X ← X \ {x'}
12     c ← c + t(a')
13 return getBestSolution(D)
```

The algorithm initializes the total running cost $c$ and the set of evaluated scenarios $D$ (lines 1-2). It then enters a loop that continues until the search budget $b_s$ is exhausted (line 3). In each iteration, the algorithm decides whether to perform random initialization or to use the model-guided search. If the current cost $c$ is less than the initialization budget $b_i$, the algorithm samples a candidate scenario $x'$ and a fidelity level $a'$ randomly from the scenario domain $X$ and the fidelity domain $A$, respectively (lines 4-6). This ensures diverse initial data for training the surrogate model. Once initialization is complete, MFBO-Drive calls the function ***getNextScenarioFidelity*** (detailed in Algorithm 2) to select the next scenario-fidelity pair $(x', a')$ (line 8). The selected scenario is then evaluated in selected fidelity using the objective function $f(x', a')$, and the result $y'$ is recorded (line 9). The new observation $(x', a', y')$ is added to the evaluation history $D$, and the scenario $x'$ is removed from the search space to to never simulate the same scenario twice (lines 10-11). The cost of the evaluation, defined by the fidelity level $a'$, is added to the running cost $c$ (line 12). Once the budget is fully consumed, the algorithm returns the best scenario discovered during the search by analyzing the evaluation history $D$ (line 13).

Algorithm 2 outlines the ***getNextScenarioFidelity*** function. It takes as input the current set of evaluated scenarios $D$,

the scenario domain $X$, the fidelity domain $A$, the fidelity exploration rate $\varepsilon$, and the fidelity error threshold $e_{max}$. It then returns a scenario-fidelity pair $(x', a')$ where $x' \in X$ is the next candidate scenario and $a' \in A$ is the fidelity level at which the evaluation should be performed.

**Algorithm 2:** ***getNextScenarioFidelity***

**Input :** Set of Evaluated Scenarios $D$, Scenario Domain $X$, Fidelity Domain $A$, Fidelity Exploration Rate $\varepsilon$, Fidelity Error Threshold $e_{max}$
**Output:** Candidate Scenario $x'$, Candidate Fidelity Level $a'$

1 Highest Fidelity Level $a^* \leftarrow$ *getHighestFidelity*$(A)$
2 Surrogate Model $(\mu, \sigma) \leftarrow$ *trainSurrogateModel*$(D)$
3 Acquisition Function $\alpha \leftarrow$ *updateAcqFunction*$(\mu, \sigma)$
4 Candidate Scenario $x' \leftarrow \text{argmax}_{x \in X}\, \alpha(x, a^*)$
5 **if** *Uniform*$(0, 1) < \varepsilon$ **then**
6  Candidate Fidelity Level $a' \leftarrow a^*$
7 **for** *Fidelity Level* $a \in$ *sort*$(A)$ **do**
8  Estimated Fidelity Error $e \leftarrow |\mu(x', a) - \mu(x', a^*)|$
9  **if** $e < e_{max}$ **then**
10  Candidate Fidelity Level $a' \leftarrow a$
11  **break**
12 **return** $(x', a')$

The algorithm begins by identifying the highest fidelity (i.e., the oracle fidelity) $a^* \in A$ (line 1) and updating the surrogate model $\mu$ based on the current history $D$ (line 2). Using the fidelity-aware model, the acquisition function $\alpha$ is updated based on $\mu$ (line 3) and maximized over $x \in X$ (at $a^*$) to select the most promising candidate scenario $x'$ (line 4). This strategy ensures that candidate selection is always driven by high-fidelity predictions. The algorithm then chooses the fidelity level $a'$ for $x'$. With an exploration probability $\varepsilon$, $a'$ is forced to $a^*$ as discussed in Section III-A (lines 5-6). Otherwise, the algorithm proceeds to search for the lowest safe fidelity. It iterates through the fidelity levels in ascending order and select the first level $a$ for which the estimated fidelity error, $|\mu(x', a) - \mu(x', a^*)|$, is below the predefined threshold $e_{max}$ (lines 7-11). Finally, the selected scenario-fidelity pair $(x', a')$ is returned (line 12).

## IV. Empirical Evaluation

Our evaluation addresses the following research questions:

**RQ1** How sensitive are the driving scenario evaluation results to changes in the fidelity parameters?

**RQ2** Which surrogate model performs best within MFBO-Drive in terms of predictive power?

**RQ3** How does MFBO compare to other scenario generation techniques in terms of effectiveness and efficiency?

**RQ4** What is the impact of varying hyperparameter values on the performance of MFBO?

**RQ1** investigates the feasibility of applying MFBO to the context of ADS testing. **RQ2** examines which surrogate model is most effective for approximating the driving evaluation function, prior to its use in an online learning setting. **RQ3** compares our multi-fidelity approach with baselines (e.g., single-fidelity BO, Random Search), in terms of effectiveness and efficiency. This is the main research question that can demonstrate the benefits of our approach. **RQ4** assesses the robustness of our approach with respect to its hyperparameters (e.g., initialization budget, and minimal probability of using the highest fidelity evaluation).

### A. Case Study Subjects

*1) Simulator:* To answer the RQs, we use *MetaDrive* [3], an actively maintained open-source driving simulator capable of realistic perception and easy scenario generation. Importantly, MetaDrive has been shown to avoid unintentional non-determinism during simulation [17], addressing a threat to the validity of search-based testing experiments. Although we use MetaDrive, a relatively fast simulator for the sake of large-scale evaluation, our approach can be applied with more computationally intensive simulators, which can take up to several hours if not days, to run thousands of simulations.

*2) ADS under Test:* For the ADS, we utilize the *expert policy*, a simple neural network-based driving agent provided by default in MetaDrive. It inputs the ray-based sensor data (Lidar, side scanner, and lane line detector) and outputs the throttle, braking, and lateral steering control. It has been pre-trained by MetaDrive's contributors, using Proximal Policy Optimization (PPO) reinforcement learning algorithm [18]. For the purpose of this evaluation, using a relatively simple ADS, such as the *expert policy*, is acceptable, since our focus is on evaluating a novel ADS-agnostic test generation framework in terms of effectiveness and efficiency.

*3) Test Scenarios:* In our evaluation, a basis of each testing scenario is a map generated by MetaDrive's built-in procedural map generator [3]. Each map is a connection of five primitive road blocks (e.g., straight road, curve, and a roundabout) that can be further parametrized (e.g., by length and the radius of curvature). The scenario generator uses a random seed to deterministically define start and end points for background vehicles. Given a defined map and background traffic configuration, the ego vehicle is tasked with navigating from a starting point (located on the first road block) to a destination (on the last road block) while satisfying safety requirements.

To enable optimization, each scenario is encoded as a fixed-length vector of real numbers, denoted as $x$. Categorical variables (e.g., block types) are encoded as integers, while continuous parameters (e.g., curvature radius) are represented directly. This vector captures the complete scenario definition, including: a road network layout, ego vehicle's initial position and destination, and the initial positions, sizes, and attributes of all background vehicles. To standardize input dimensions, we fix the maximum number of background vehicles at 40, resulting in a vector of length 461. Scenario vectors containing fewer vehicles are padded with placeholder values (i.e., $-1$) to fill the missing fields.

*4) Safety Requirements:* For each scenario, we evaluate the ADS by counting the number of infractions for the following safety requirements in MetaDrive: (1) reach the destination within the time limit; (2) no collisions with other vehicles; (3) no collisions with sidewalks; (4) no out-of-lane episodes.

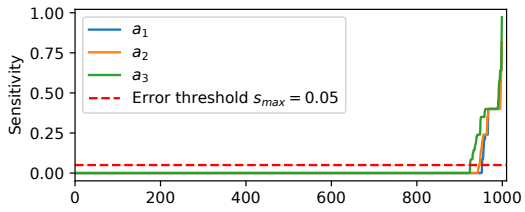


Fig. 1. Results of sensitivity analysis for a 1000 driving scenarios, across three lower fidelity configurations in $A_{eval}$.

We selected the above requirements as they capture the most important requirements of safe autonomous driving.

The objective function $f : X \rightarrow \mathbb{R}$ quantifies scenario criticality using the *Driving Score* (DScore), adapted from the CARLA Leaderboard[2]. Specifically, DScore $= R_c \times R_d \times 0.65^{n_{sw}} \times 0.60^{n_{veh}}$, where $R_c$ is the percentage of route completed, $R_d$ is the percentage of time driven in correct lane, $n_{sw}$ is the number of collisions with sidewalks, and $n_{veh}$ is the count of collisions with other vehicles. It captures the overall driving performance of the ADS in a scenario, ranging from 1 (no infractions) to 0 (full of infractions).

*5) Fidelity Levels:* For evaluation, we use a discrete set of fidelity levels $A_{eval} = \{10 \text{ FPS}, 20 \text{ FPS}, 30 \text{ FPS}, 60 \text{ FPS}\}$ by varying FPS in MetaDrive. Our preliminary study confirmed that simulation cost is linearly dependent with the FPS parameter: running at 20 FPS costs twice as much as at 10 FPS, and so on. To simplify notation, we define the execution unit cost as $c(a) = \frac{a}{10}$. Thus, a 10 FPS simulation has a cost of 1, a 20 FPS simulation a cost of 2, and so forth.

### B. RQ1: Effects of fidelity parameter on driving evaluation

*1) Setup:* To address RQ1, we conducted a sensitivity analysis to assess how the driving scenario evaluation results vary with changes in the fidelity parameter. Given an ordered set of fidelity parameters $A = \{a_1 = 10 \text{ FPS}, a_2 = 20 \text{ FPS}, a_3 = 30 \text{ FPS}, a_4 = 60 \text{ FPS}\}$, the sensitivity of a scenario $x$ at a fidelity level $a_i$ for $i \in \{1, 2, 3\}$ is defined as $Sen(x, a_i) = |f(x, a_i) - f(x, a_{i+1})|$ where $f(x, a_i)$ denotes the DScore of the ADS when evaluated on $x$ under some lower fidelity $a_i$.

Intuitively, for MFBO to be effective, $Sen(x, a_i)$ should remain below a small threshold $s_{max}$ for most $x \in X$ and $a_i \in A$. Otherwise, if $Sen(x, a_i)$ is high across many $x$ and $a_i$, even small changes in fidelity may lead to large variations in the DScore of $x$, undermining the assumption that lower-fidelity simulation results provide reliable approximations of higher-fidelity results. To evaluate if $Sen(x, a_i)$ is smaller than $s_{max} = 0.05$ (meaning 5% of DScore) for most $x$ and $a_i$, we randomly generated a set of 1,000 driving scenarios $X_{1k}$ using MetaDrive's built-in map generator and calculated $Sen(x, a_i)$ for all $x \in X_{1k}$ and $a \in A_{eval}$.

*2) Results:* Figure 1 shows the distribution of sensitivity of 1,000 driving scenarios under different fidelity levels, sorted by magnitude. Notably, 92.5% of scenarios fall below $s_{max}$, indicating that their evaluation results remain consistent regardless of simulation fidelity. This is not just due to the triviality of the scenarios; they involve diverse driving tasks, including lane changes, turns, and interactions with other vehicles. Interestingly, our inspection reveals no apparent structural characteristics (e.g., specific road shapes) that distinguish fidelity-sensitive scenarios from insensitive ones. This lack of apparent patterns further highlights the necessity of our MFBO-Drive approach, which can automatically model the complex, non-linear relationship between scenario parameters, simulation fidelity, and safety outcomes.

> The answer to RQ1 is that 92.5% of driving scenarios are unaffected by changes in simulation fidelity, showing a strong potential for applying multi-fidelity optimization.

### C. RQ2: Surrogate Models

*1) Setup:* To answer RQ2, we compare surrogate models by training them on the same dataset and evaluating their prediction accuracy, training time, and inference time (i.e., time to make a prediction). As described in Section III, we adopt an input-augmented surrogate model that predicts the fitness value $f(x, a)$ given a scenario definition $x$ and fidelity setting $a$. For training, we reused the dataset collected in RQ1 (Section IV-B). Specifically, we sampled 1,000 scenario definitions from MetaDrive, each representing the simulator's initial state at the start of a scenario. Every scenario was evaluated under all fidelity settings in the evaluation set $A_{eval}$, resulting in a complete dataset of 4,000 evaluation results $f(x, a)$.

To evaluate potential surrogate modeling approaches, we evaluated a handful of popular machine learning techniques, provided in Scikit-learn [19]: (1) Gaussian Process regression, a traditional approach in the field, Linear models (2) Ridge, (3) Lasso, (4) ElasticNet, and following ensemble: (5) Random Forest, (6) Gradient Boosting. During the evaluation, we did not tweak the default settings, nor did we perform hyperparameter tuning.

To evaluate model performance, we conducted 10-fold cross-validation using randomized splits, repeated across ten different random seeds. To assess predictive accuracy, we measured the coefficient of determination ($R^2$), a standard metric for evaluating the performance of regression models. In addition to prediction accuracy, we recorded the total training time and total prediction (inference) time for each model across all folds and seeds. Final results were obtained by averaging these metrics over the ten seeds to ensure robustness and reduce variance due to random initialization.

*2) Results:* Table I summarizes the performance results for the evaluated regression methods. Most notably, Gaussian Process regression performed poorly, achieving a negative $R^2$ score of -20.723, meaning that the trained model did not possess any predictive power. This could be mainly because GP works best with vectors of continuous features [11] of low dimensionality ($|x| < 20$), while our driving scenarios contain a large number of categorical features. Linear models turned out to be quick to train (less than half a second) and managed to achieve some level of predictive power. Surprisingly, the Ridge linear model achieved a satisfactory $R^2$ score of 0.4.

[2]https://leaderboard.carla.org

TABLE I
SURROGATE MODEL PERFORMANCE STATISTICS

| Regression Method | Method Type | $R^2$ | | Pred Time [s] | | Train Time [s] | |
|---|---|---|---|---|---|---|---|
| | | Mean ↓ | Std | Mean | Std | Mean | Std |
| Random Forest | Ensemble | **0.833** | 0.044 | 0.049 | 0.007 | 16.612 | 1.134 |
| Gradient Boost | Ensemble | 0.604 | 0.047 | 0.044 | 0.005 | 6.726 | 0.433 |
| Ridge | Linear | 0.402 | 0.068 | 0.046 | 0.006 | 0.445 | 0.032 |
| Elastic Net | Linear | 0.075 | 0.023 | 0.047 | 0.006 | 0.446 | 0.032 |
| Lasso | Linear | 0.043 | 0.016 | 0.048 | 0.008 | 0.446 | 0.035 |
| GP | Other | -20.723 | 3.150 | 0.760 | 0.400 | 7.498 | 1.009 |

(↓ The table is sorted by the mean $R^2$ value in descending order)

The ensemble models outperforms, achieving $R^2$ scores equal to $0.60$ and $0.83$ respectively. They also showed quick inference (prediction) time of less than $0.05s$, making them a good fit for Bayesian Optimization. While the ensemble methods incurred higher training times than linear models, this overhead is negligible in the broader context of ADS testing, where simulation-based evaluations are considerably more computationally expensive.

The answer to RQ2 is that Random Forest surrogate models offer the best prediction accuracy while maintaining high efficiency, making a suitable choice for MFBO-Drive.

### D. RQ3: Efficiency and effectiveness of MFBO

*1) Setup:* To address RQ3, we assessed the effectiveness and efficiency of our MFBO approach, configured with the best-performing surrogate model identified in RQ2. We compared MFBO against two baseline methods: Random Search (RS) and Single-Fidelity Bayesian Optimization (SFBO). RS is a common, robust baseline for search-based testing [6], while SFBO serves as a meaningful comparator since it shares the same Bayesian optimization framework and surrogate model as MFBO but operates at a single fidelity level. This comparison isolates the contribution of multi-fidelity-based optimization in the context of ADS testing.

All approaches were allocated the same total budget of 600 simulation time units, equivalent to 100 high-fidelity evaluations or 600 low-fidelity evaluations. For both SFBO and MFBO, 10% of the budget (i.e., 60 time units) was reserved for the random initialization phase (i.e., $b_i = 0.1b_s$). Both used the Random Forest surrogate from as identified in RQ2. The predictive mean $\mu(x)$ was computed as the average output of 100 random trees, and the predictive uncertainty $\sigma(x)$ as the standard deviation of their outputs. For acquisition, we used the Upper Confidence Bound (UCB) strategy (see Section II-B). RQ4 will further examine the impact of varying $b_i$ values.

To isolate the benefit of multi-fidelity optimization, RS and SFBO were limited to the highest (oracle) fidelity. MFBO, in contrast, was permitted to dynamically select lower fidelities when predicted to be safe, following the strategy outlined in Algorithm 2. The hyperparameter $\varepsilon = 0.1$ guarantees that at least 10% of the evaluations are performed at the highest fidelity. RQ4 will further examine the effect of varying $\varepsilon$ values.

For a fair comparison across approaches, we constructed a diverse benchmark set of 100,000 driving scenarios $X_{100k}$ using MetaDrive's built-in map generator. To find the global optimum (i.e., the scenario with the lowest DScore) among $X_{100k}$, we executed every scenario $x \in X_{100k}$ once at the highest fidelity level $a^*$, yielding its oracle DScore $f(x, a^*)$. The scenario with the lowest oracle DScore is denoted $x^* \in X_{100k}$.

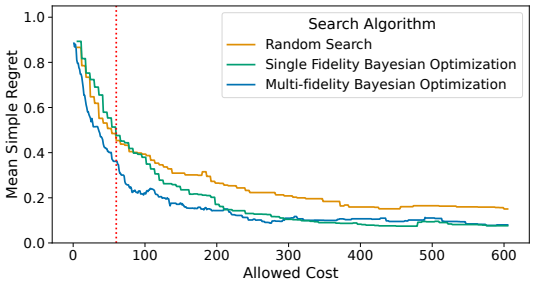


Fig. 2. Search convergence for scenario generation, visualized as a mean simple regret over time for each search approach[4]. The red dotted line indicates the end of the initialization phase.

We quantify the effectiveness of each testing approach using simple regret, which measures the deviation from the global optimum $x^*$ after consuming a given computation budget. Specifically, for the best scenario $x_b$ identified by a search algorithm after spending budget $b$, the simple regret is defined as $r(x_b) = |f(x_b, a^*) - f(x^*, a^*)|$, where $f(x, a^*)$ denotes the objective value evaluated at the oracle fidelity level $a^*$. This metric captures how close the best-found solution is to the true optimum, providing a direct indicator of search quality.

Due to the inherent randomness of all approaches (e.g., random initialization, surrogate model learning, scenario sampling), each experiment was repeated 50 times. To evaluate performance differences, we used the following statistical methods: (1) the *Mann-Whitney U test*, to assess whether MFBO significantly outperforms each baseline in terms of simple regret; (2) the $A_{12}$ *effect size*, to quantify the probability that MFBO yields lower regret than the comparator.

*2) Results:* Figure 2 illustrates the relationship between the budget consumed and simple regret across evaluated approaches, while Table II presents the statistical test results.

Among the approaches, RS consistently exhibits the highest regret values throughout the budget range, indicating the weakest performance in both effectiveness and efficiency. In contrast, MFBO-Drive achieves the lowest regret across nearly all budget levels, reflecting superior cost-effectiveness, as well as the lowest final regret at the end of the search, indicating stronger overall effectiveness.

When comparing MFBO-Drive and RS, MFBO-Drive achieves a 36.7% improvement in cost-effectiveness, measured by the area under the simple regret curve (AUC). Even during the initialization (the first 10% of the budget, marked by the red dashed line in Figure 2), MFBO-Drive benefits from low-cost, low-fidelity evaluations, yielding lower regret than RS, which relies solely on high-fidelity sampling. After initial-

[4]By definition, the curve of simple regret for high-fidelity only approaches should be monotonically decreasing. However, due to simulator nondeterminism (as discussed in Section IV-F2), the evaluation of $f(x, a^*)$ may yield different results.

TABLE II
STATISTICAL COMPARISON OF MFBO-DRIVE AGAINST RANDOM SEARCH (RS) AND SINGLE-FIDELITY BAYESIAN OPTIMIZATION (SFBO)

| Compared Algorithms | Δ Cost-Effectiveness | At time: | 100 | 200 | 300 | 400 | 500 | 600 |
|---|---|---|---|---|---|---|---|---|
| MFBO, RS | **36.73%** | $A_{12}$ | 0.703 | 0.697 | 0.715 | 0.665 | 0.656 | 0.693 |
| | | p-value | 0.000 | 0.001 | 0.000 | 0.004 | 0.007 | 0.001 |
| MFBO, SFBO | **16.84%** | $A_{12}$ | 0.677 | 0.597 | 0.533 | 0.489 | 0.479 | 0.475 |
| | | p-value | 0.002 | 0.094 | 0.568 | 0.847 | 0.722 | 0.674 |

(Δ Cost-Effectiveness: difference in cost-effectiveness measured by the area under the simple regret curve, $A_{12}$: effect sizes, p-value: Mann-Whitney U test p-value)

ization, MFBO-Drive maintains a 22.2% lower simple regret than RS, reinforcing its advantage in early-stage performance.

In comparing MFBO-Drive and SFBO, MFBO-Drive outperforms SFBO during the first half of the search (i.e., up to 300 units), using low-fidelity evaluations to quickly identify promising regions. However, this advantage diminishes in the second half, where both approaches converge toward near-optimal solutions and exhibit similar final performance. Mann-Whitney U tests confirm that there is no significant difference in final effectiveness between MFBO-Drive and SFBO (p-value $> 0.5$; see Table II). Nonetheless, MFBO gains a 16.8% improvement in cost-effectiveness over SFBO, based on AUC. This demonstrates that MFBO makes more efficient use of the evaluation budget without sacrificing solution quality.

The answer to RQ3 is that MFBO improves early-stage efficiency without compromising the final solution quality, making it a practical and cost-effective alternative to single-fidelity methods in budget-constrained ADS testing.

### *E. RQ4: Impact of hyperparameters*

*1) Setup:* To assess the effect of MFBO-Drive's hyperparameters, we ran experiments varying (1) the fidelity exploitation rate $\varepsilon$, and (2) the initialization budget $b_i$, while using the same setup as the baseline comparison. In the first, we varied $\varepsilon$ from 0 to 0.5 in steps of 0.05, while keeping the initialization budget fixed at $b_i = 10\%\, b_s$, where $b_s = 600$ is the total search budget. In the second, we varied $b_i$ from 5% to 20% in steps of 5%, while keeping $\varepsilon = 0.1$. We evaluated each variant's efficiency and effectiveness, using the same metrics as in RQ3, comparing them against a baseline configuration of MFBO with $\varepsilon = 0.1$ and $b_i = 10\%$.

*2) Results:* Table III shows the comparison between the baseline MFBO-Drive and MFBO-Drive's variants with varied values of hyperparameters: $\varepsilon$ and $b_i$. Full results for tested $\varepsilon$ and $b_i$ values are in the replication package (Section IV-F3).

Overall, we observe that the effectiveness of MFBO-Drive is robust to variations in $\varepsilon$. Pairwise Mann-Whitney U tests reveal no statistically significant differences in final simple regret across different $\varepsilon$ settings (lowest p-value: 0.262). In terms of efficiency, however, configurations with higher $\varepsilon$ values (e.g., $\varepsilon$ = 0.5) exhibit poorer performance relative to the baseline. This is expected, as evaluating at the highest fidelity too frequently (more than 50% of the time) diminishes the advantage of cost-effective optimization through multi-fidelity modeling. Although there is no statistically significant

TABLE III
STATISTICAL COMPARISON OF MFBO-DRIVE AGAINST MFBO-DRIVE VARIANTS WITH NON-DEFAULT HYPERPARAMETERS

| Compared Algorithms | Δ Cost-Effectiveness | At time: | 100 | 200 | 300 | 400 | 500 | 600 |
|---|---|---|---|---|---|---|---|---|
| MFBO, $\text{MFBO}_{\varepsilon=0.0}$ | 8.42% | $A_{12}$ | 0.504 | 0.459 | 0.436 | 0.444 | 0.460 | 0.565 |
| | | p-value | 0.950 | 0.485 | 0.269 | 0.335 | 0.489 | 0.262 |
| MFBO, $\text{MFBO}_{\varepsilon=0.2}$ | -1.05% | $A_{12}$ | 0.497 | 0.487 | 0.467 | 0.456 | 0.435 | 0.482 |
| | | p-value | 0.959 | 0.830 | 0.568 | 0.447 | 0.263 | 0.759 |
| MFBO, $\text{MFBO}_{\varepsilon=0.5}$ | -22.11% | $A_{12}$ | 0.587 | 0.579 | 0.525 | 0.493 | 0.441 | 0.479 |
| | | p-value | 0.132 | 0.173 | 0.669 | 0.904 | 0.312 | 0.720 |
| MFBO, $\text{MFBO}_{b_i=5\%}$ | -1.34% | $A_{12}$ | 0.490 | 0.533 | 0.541 | 0.505 | 0.504 | 0.570 |
| | | p-value | 0.863 | 0.569 | 0.479 | 0.939 | 0.945 | 0.227 |
| MFBO, $\text{MFBO}_{b_i=15\%}$ | 1.48% | $A_{12}$ | 0.536 | 0.422 | 0.481 | 0.460 | 0.436 | 0.495 |
| | | p-value | 0.539 | 0.180 | 0.746 | 0.496 | 0.269 | 0.939 |
| MFBO, $\text{MFBO}_{b_i=20\%}$ | -7.71% | $A_{12}$ | 0.553 | 0.585 | 0.512 | 0.473 | 0.495 | 0.550 |
| | | p-value | 0.361 | 0.145 | 0.839 | 0.638 | 0.937 | 0.385 |

(Δ Cost-Effectiveness: difference in cost-effectiveness measured by the area under the simple regret curve, $A_{12}$: effect sizes, p-value: Mann-Whitney U test p-value)

difference between configurations with $\varepsilon = 0.0$ and $\varepsilon = 0.1$, we observed that setting $\varepsilon = 0.0$ can cause instability in some search runs, resulting in a slight increase in simple regret. Therefore keeping $\varepsilon = 0.1$ is preferred in practice due to its greater robustness and more reliable convergence behavior.

Table III shows that MFBO-Drive is robust to changes in the initialization budget: we observed no statistically significant differences in performance between runs with different amounts of budget allocated to the initial random sampling (lowest p-value: 0.227). This robustness arises from the nature of the early search phase. When the surrogate model has access to only limited data, the acquisition function tends to assign similar values to most candidate scenarios. As a result, MFBO-Drive behaves similarly to random sampling during this phase, regardless of the exact value of $b_i$, thereby ensuring reasonably unbiased exploration of the search space.

The answer to RQ4 is that MFBO's efficiency is generally robust to the choice of $\varepsilon$ and $b_i$. However, we observe that smaller values of $\varepsilon$ tend to yield slightly higher efficiency by reducing the number of high-cost evaluations.

### *F. Threats to Validity*

*1) External validity:* We use a specific simulator (i.e., MetaDrive) with specific driving agent (i.e., expert policy) that poses an external validity threat. A different simulator paired with a more advanced driving agent is a subject to future experiments. Reproducing the same evaluation would require 230 thousand driving simulations, given that a single scenario execution takes a minute on average, it would require 160 computing days to complete. Nonetheless, further experiments with other ADSs and simulators are needed to demonstrate generalizability. MFBO is not limited to any particular simulator, most modern ones (e.g., CARLA [4]) expose fidelity parameters such as frame-rate or rendering settings.

In our methodology, we assumed that the cost of simulation depends solely on the fidelity parameter. In reality, cost can also vary with scenario itself. To give an example, if driving agent gets blocked with other traffic vehicle, simulation will take a longer time. Lastly, there could be external factors that add the potentially non-negligible constant time, e.g., restarting the driving simulator between evaluations.

*2) Internal validity:* A big threat to validity is non-determinism of driving simulation [17] making reproducibility challenging. To minimise this, we adopted following mitigation strategies: we seeded all relevant random number generators and restarted the simulation environment after each scenario execution to ensure consistent initialization. In our evaluation, we executed 9729 driving scenarios twice in the highest fidelity. We found that 1.52% of these scenarios produced diffrent results on re-evaluation (when compared to five significant digits), indicating a low but present level of unintentional nondeterminism.

*3) Data availability:* The replication package is available on Figshare https://figshare.com/s/ba8790c8ad504514be41

## V. Related work

Our research sits at the intersection of black-box testing for Autonomous Driving Systems (ADS) and applied multi-fidelity optimization. Black-box ADS testing seeks test inputs (i.e., a driving scenario) that expose a safety requirement violation (e.g., a collision). Recent studies have accelerated critical scenario generation with many search-based methods in particular surrogate-assisted testing [6, 20] which leverages machine-learning models to capture complex input-output landscape of the underlying objective function to help navigating the search domain. Abeysirigoonawardena et al. [13] applied Bayesian Optimization to find scenario parameters of dynamic entities (i.e., cars and pedestrians) that cause safety violation of the ADS under test. Multi-Fidelity Bayesian Optimization (MFBO), extends above methods giving ways for the strategic selection of physics modeling settings (i.e., the fidelity) trying to achieve the balance between precision and efficiency [14]. This is especially beneficial when acquiring high-accuracy data is costly or computationally intensive. MFBO has been successfully applied in various engineering tasks, such as neural network hyperparameter tuning [21] and engineering design [15, 22]. However, no existing study addresses challenges of applying multi-fidelity optimization to the problem of driving scenario generation for ADS testing.

## VI. Conclusion

This paper introduced MFBO-Drive, a novel approach for cost-effective critical scenario generation in scenario-based testing of ADS. By formulating the problem as a multi-fidelity optimization problem, MFBO-Drive enables to dynamically select both driving scenarios and the corresponding fidelity levels for evaluation. Empirical evaluation using the MetaDrive simulator shows that MFBO-Drive significantly outperforms both single-fidelity Bayesian Optimization and Random Search in terms of cost-effectiveness. It demonstrates that MFBO-Drive is well-suited for budget-limited testing environments. As part of future work, we will explore the application of MFBO-Drive to state-of-the-art ADS and more computationally expensive simulators.

## References


[1] X. Zhang, J. Tao, K. Tan, M. Törngren, J. M. G. Sánchez, M. R. Ramli, X. Tao, M. Gyllenhammar, F. Wotawa, N. Mohan, M. Nica, and H. Felbinger, "Finding critical scenarios for automated driving systems: A systematic mapping study," *IEEE Transactions on Software Engineering*, vol. 49, no. 3, pp. 991–1026, 2023.

[2] W. Ding, C. Xu, M. Arief, H. Lin, B. Li, and D. Zhao, "A survey on safety-critical driving scenario generation—a methodological perspective," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 7, pp. 6971–6988, 2023.

[3] Q. Li, Z. Peng, L. Feng, Q. Zhang, Z. Xue, and B. Zhou, "Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 45, no. 3, pp. 3461–3475, 2022.

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

[5] Z. Zhong, G. Kaiser, and B. Ray, "Neural network guided evolutionary fuzzing for finding traffic violations of autonomous vehicles," *IEEE Transactions on Software Engineering*, vol. 49, no. 4, pp. 1860–1875, 2023.

[6] F. U. Haq, D. Shin, and L. Briand, "Efficient online testing for dnn-enabled systems using surrogate-assisted and many-objective optimization," in *Proc. of the 44th International Conference on Software Engineering*, ser. ICSE '22, 2022, p. 811–822.

[7] D. Huang, T. T. Allen, W. I. Notz, and R. A. Miller, "Sequential kriging optimization using multiple-fidelity evaluations," *Structural and Multidisciplinary Optimization*, vol. 32, pp. 369–382, 2006.

[8] K. Kandasamy, G. Dasarathy, J. Schneider, and B. Póczos, "Multi-fidelity Bayesian optimisation with continuous approximations," in *Proc. of the 34th International Conference on Machine Learning*, ser. Proc. of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 1799–1808.

[9] Z. Zhong, Y. Tang, Y. Zhou, V. d. O. Neves, Y. Liu, and B. Ray, "A survey on scenario-based testing for automated driving systems in high-fidelity simulation," 2021.

[10] M. Klischat and M. Althoff, "Generating critical test scenarios for automated vehicles with evolutionary algorithms," in *Proc. of the IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 2352–2358.

[11] P. I. Frazier, "A tutorial on bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.

[12] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[13] Y. Abeysirigoonawardena, F. Shkurti, and G. Dudek, "Generating adversarial driving scenarios in high-fidelity simulators," in *ICRA*, 2019.

[14] K. Li and F. Li, "Multi-fidelity methods for optimization: a survey," *arXiv:2402.09638*, 2024.

[15] B. Do and R. Zhang, "Multifidelity bayesian optimization: A review," *AIAA Journal*, pp. 1–37, 2023.

[16] M. Meliani, N. Bartoli, T. Lefebvre, M.-A. Bouhlel, J. R. Martins, and J. Morlier, "Multi-fidelity efficient global optimization: Methodology and application to airfoil shape design," in *AIAA aviation 2019 forum*, 2019, p. 3236.

[17] O. Osikowicz, P. McMinn, and D. Shin, "Empirically evaluating flaky tests for autonomous driving systems in simulated environments," in *2025 IEEE/ACM International Flaky Tests Workshop (FTW)*. Institute of Electrical and Electronics Engineers (IEEE), 2024.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv:1707.06347*, 2017.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[20] S. Nejati, L. Sorokin, D. Safin, F. Formica, M. M. Mahboob, and C. Menghi, "Reflections on surrogate-assisted search-based testing: A taxonomy and two replication studies based on industrial adas and simulink models," *Information and Software Technology*, 2023.

[21] A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter, "Fast bayesian optimization of machine learning hyperparameters on large datasets," in *Proc. of the 20th International Conference on Artificial Intelligence and Statistics*, 2017, pp. 528–536.

[22] S. Li, W. Xing, R. Kirby, and S. Zhe, "Multi-fidelity bayesian optimization via deep neural networks," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, 2020, pp. 8521–8531.