# On the Effectiveness of Open-Loop DNN Repair in End-to-End Autonomous Driving Systems

Deyun Lyu
*National Institute of Informatics*
Tokyo, Japan
lyudeyun@nii.ac.jp

Paolo Arcaini
*National Institute of Informatics*
Tokyo, Japan
arcaini@nii.ac.jp

Zhenya Zhang
*Kyushu University and*
*National Institute of Informatics*
Fukuoka, Japan
zhang@ait.kyushu-u.ac.jp

Masaaki Inoue
*Toyota Motor Corporation*
Tokyo, Japan
masaaki_inoue_aa@mail.toyota.co.jp

Nayuta Yanagisawa
*Toyota Motor Corporation*
Tokyo, Japan
nayuta_yanagisawa@mail.toyota.co.jp

Ryo Neyama
*Toyota Motor Corporation*
Tokyo, Japan
neyama@mail.toyota.co.jp

Shintaro Fukushima
*Toyota Motor Corporation*
Tokyo, Japan
s_fukushima@mail.toyota.co.jp

Fuyuki Ishikawa
*National Institute of Informatics*
Tokyo, Japan
f-ishikawa@nii.ac.jp

***Abstract*—End-to-End (E2E) autonomous driving systems integrate perception, prediction, and planning into a unified DNN model. While the integrated architecture brings some benefits, these models can still make unsafe planning decisions. Once such faulty behaviors are observed, retraining the entire model is often costly and yields limited improvement. DNN repair has been proposed as a fine-grained alternative that uses a search algorithm to adjust some faulty model parameters to fix unsafe behaviors without full retraining. In this paper, we present a practical experience report on the application of DNN repair for E2E models. In principle, during repair, the performance of each candidate repaired model should be assessed in a simulation environment (i.e., in a closed-loop manner) to check that the vehicle does not encounter any unsafe situation. However, such assessment is too costly due to the long simulation time and, so, it does not allow to repair the model in a scalable way. Therefore, we investigate whether using open-loop metrics that check the quality of single inferences of the E2E model (which are much faster to compute) allows to effectively improve the model. To this aim, we propose a novel open-loop DNN repair approach for trajectory planning in E2E models. In the approach, we introduce a way to split repair data between correct and wrong data based on two open-loop metrics, i.e., L2 error and number of collisions. Moreover, we introduce three alternative fitness functions that make use of the two open-loop metrics in the assessment of the candidate repaired models. The approaches have been evaluated on two representative E2E architectures, i.e., UNIAD and VAD. Key lessons from experiments include: our proposed approaches can improve the models under repair in terms of open-loop performance; however, these improvements are not always reflected in improvements on closed-loop performance.**

***Index Terms*—DNN repair, fault localization, E2E ADS, open-loop metrics, closed-loop metrics**

P. Arcaini is supported by the ASPIRE grant No. JPMJAP2301, JST. Z. Zhang is supported by JST BOOST Grant Number JPMJBY24D7, and JSPS KAKENHI Grant Number JP25K21179.

## I. INTRODUCTION

Autonomous Driving Systems (ADSs) promise to bring several advantages in the transportation domain, such as a reduced number of accidents and traffic congestion. The typical ADS architecture is a modular pipeline [1], where the whole driving task is decomposed in specific tasks (e.g., perception, prediction, planning) performed by specialized components communicating with each other. Despite the advantages like better interpretability, the modular architecture suffers from some issues: (i) errors from different components can propagate and amplify through the pipeline, (ii) the communication among the different components causes a computational burden, and (iii) while the single components optimize for their own task, they do not optimize for the final goal of planning.

A new type of ADS architecture, End-to-End (E2E) ADS [2] is emerging as a solution to the aforementioned issues, through the adoption of E2E DNN models. Since the different tasks are combined in a single model, the whole system is optimized for the final goal of planning; moreover, the monolithic nature maximizes the computational efficiency. Modern E2E models, such as UNIAD [3] and VAD [4], adopt a transformer-based architecture to jointly model perception, prediction, and planning in a unified framework, enabling direct mapping from sensor inputs to planned trajectories. Thanks to their advantages, E2E models have attracted attention from industry, including our industrial partner from the automotive domain.

However, developing E2E models that behave safely in all driving situations is impossible due to (i) their data-driven nature and (ii) the emergence, at operation time, of driving situations not represented in the training set. A possible

solution to improve the safety of an E2E ADS is to perform additional training to fix the observed misbehaviors. However, retraining is not only costly, but can also introduce unintended degradation on previously correct behaviors [5].

In recent years, *DNN repair* [6]–[11] has established itself as an approach to perform targeted improvements of a DNN, without degrading previously correct behaviors. One of the main paradigms in this domain works in two phases. A *fault localization* phase first identifies few components of the DNN (e.g., weights) that are responsible for wrong behaviors; then, a *search-based repair* phase (using a search algorithm like Differential Evolution) is used to search for an alternative configuration of these components (e.g., new values for the weights) that allows to obtain better performance.

In this practical experience report, motivated by the success of DNN repair, we describe our efforts in developing E2EREP, a novel repair approach for E2E models. E2EREP follows the repair paradigm described above (i.e., fault localization followed by search-based repair), but introduces specific novelties to address the challenges presented by E2E models.

Search-based repair must be guided by a fitness function that uses some *evaluation metric* assessing the reliability of the candidate repaired E2E models. However, finding suitable evaluation metrics is a challenge. The best way would be to evaluate the model at the "system-level", i.e., execute the E2E model in a *closed-loop* simulation (using an ADS simulator like CARLA) over driving scenarios as those provided by `Bench2Drive` [12], and measure *closed-loop* metrics (e.g., the *driving score*) that assess the overall safety and the performance of the autonomous vehicle. However, using these closed-loop metrics is infeasible and not applicable in a real industrial context. Indeed, even using a powerful GPU machine, each simulation of an E2E model over a driving scenario can take tens of minutes; since a repair approach must evaluate multiple candidate models, the whole repair would take days.

Therefore, in this paper, we investigate whether it is possible to effectively repair an E2E model by evaluating it at the "component level", using *open-loop* metrics that only consider single inferences of the model and do not require expensive simulations. Typical open-loop metrics are the deviation of the planned trajectory from the ground truth trajectory (*L2 error*), or the number of predicted collisions.

Existing DNN repair approaches are not directly applicable to the repair of E2E models using open-loop metrics. Indeed, most of these approaches are applicable only to classification models [6]–[11], for which there is a clear separation between *positive* and *negative* data, i.e., data that evaluate correctly and not correctly, respectively; open-loop metrics, instead, are quantitative metrics for which there is no such clear separation (e.g., what is an acceptable *L2 error* is difficult to decide). Moreover, existing DNN repair approaches have been applied to classical DNN models, but they have not considered more complex architectures like transformers used in E2E models.

To address the previous challenges, we developed E2EREP as follows. First of all, we improve the way of splitting positive and negative data in fault localization. The only available approach in the literature to do such splitting, has been proposed by Schneider et al. [13] for semantic segmentation models, that simply splits the data in half according to an error metric like L2 error. Such approach has the drawback that it could consider, depending on the error distribution, inputs with very low L2 error as negative, or inputs with very high L2 error as positive. Therefore, in this paper, we propose a more accurate method that considers the *impact* of the error on the driving performance: inputs that can potentially lead to collisions are first labeled as negative; among the remaining inputs, those with high L2 error are also labeled as negative, while those with low L2 error are considered as positive; all the other inputs are discarded from fault localization, as they are not indications of clearly good or bad driving situations and, so, cannot help in discriminating faulty components of the E2E model. Given the obtained split data, we perform fault localization by selecting a set of *suspicious weights*, i.e., those that contribute the most to failures.

Given the identified suspicious weights, we use a search algorithm to find an alternative configuration for them that leads to better performance of the E2E model. As explained previously, to assess the performance, we cannot rely on closed-loop simulation. Thus, we investigate the adoption of open-loop metrics; we propose three novel fitness functions that differ in the open-loop metrics used to guide the search: (i) $fit_{ER}$ only considers the cumulative L2 error; (ii) $fit_{ERC}$ considers the total number of collisions and the cumulative L2 error; (iii) $fit_{CA}$ considers the total number of collisions and how many inputs have the L2 error above and below given thresholds (but not the specific values of the L2 error).

We experimented with six versions of E2EREP over two representative E2E architectures, i.e., UNIAD and VAD. Results show that our newly proposed fault localization approach is more effective than the baseline fault localization approach. Moreover, experiments show that different versions of E2EREP can consistently improve the model under repair; however, the fitness function that provides the best guidance depends on the specific model and metric of interest. However, experiments also show that the better open-loop performance does not always correspond to a better closed-loop performance, which calls for more research in this direction.

To summarize, our contributions are as follows:

(1) We propose E2EREP, an open-loop repair approach that combines fault localization and search-based weight repair for E2E models. In particular, E2EREP introduces a novel data splitting strategy based on predicted collisions and L2 error to support fault localization.
(2) We design and compare three fitness functions based on L2 error and predicted collisions to guide repair without requiring expensive closed-loop simulations.
(3) We evaluate E2EREP on two E2E ADS models, UNIAD and VAD, showing that open-loop repair can yield better open-loop behavior, but such improvements do not always lead to better closed-loop performance.

## II. Preliminaries

We here provide the minimal background notions necessary to understand the proposed approach.

### A. E2E ADS and Models

E2E ADSs rely on large neural networks to perform trajectory planning directly as output. The model integrates sensor observations, navigation goals, and vehicle states to infer a short-horizon planned trajectory for each input frame. This trajectory is one model inference and is subsequently executed by a low-level controller, to generate the control commands.

The generated trajectories correspond to future motion decisions of the vehicle. Thus, although an E2E ADS is realized as a single learning-enabled model, its trajectory generation can be understood as producing planning-level outputs in the driving pipeline.

E2E models take as input *multimodal* information perceived from sensors and maps. Such data is processed by a series of neural networks such as convolutional layers and transformers, to produce the final planning trajectory.

**Definition 1** (E2E model)**.** An E2E model is a transformer-based model $\mathcal{M}$ that processes an *input frame* $d$ (*input* in the following) composed of images taken from multiple cameras, LiDAR data, radar data, and navigation data. As output, it produces a planned trajectory $\mathbf{o}$, i.e., $\mathcal{M}(d) = \mathbf{o}$. We identify with $\mathcal{M}.W$ the *weights* of the model. The weights decide how the model maps inputs to outputs. The weights are decided by model training on a training dataset.

### B. Open-loop evaluation metrics

In order to evaluate the quality of E2E models, a possible solution is to use *open-loop* metrics that assess the quality of the outputs produced by the E2E model for each single inference. They differentiate from *closed-loop* metrics that evaluate the execution of the E2E model in a simulation environment and will be introduced in §IV-G.

Among the available open-loop metrics, we elaborate on two commonly-used ones [12], *L2 error* and *collisions*.

In our context, we assume to have a labeled dataset $D$ of pairs $(d, \hat{\mathbf{o}})$, i.e., inputs $d$ with their associated ground truth planned trajectories $\hat{\mathbf{o}}$.

Given an input $d$ and a model $\mathcal{M}$, we compute the *L2 error* as the $L^2$ norm between the planned trajectory $\mathcal{M}(d) = \mathbf{o}$ and the ground truth trajectory $\hat{\mathbf{o}}$, i.e.,

$$\texttt{L2\_Err}(\mathcal{M}, d) = \|\mathbf{o} - \hat{\mathbf{o}}\|_2 \tag{1}$$

Given a dataset $D$ and a model $\mathcal{M}$, we compute the *mean L2 error* as:

$$\mu_{\texttt{L2\_Err}}(\mathcal{M}, D) = \frac{1}{|D|} \sum_{d \in D} \texttt{L2\_Err}(\mathcal{M}, d) \tag{2}$$

Moreover, we adopt another common metric for E2E models that checks whether the planned trajectory can lead to possible *collisions*. For a given model $\mathcal{M}$ and input $d$, we define a predicate to evaluate whether the planned trajectory at that frame (i.e., the trajectory produced by a single inference) overlaps with obstacles within its 3-second prediction horizon:

$$\texttt{isCollision}(\mathcal{M}, d) = \begin{cases} true & \text{the trajectory } \mathbf{o} \text{ overlaps with obstacles in 3 secs} \\ false & \text{otherwise} \end{cases} \tag{3}$$

Given a dataset $D$ and a model $\mathcal{M}$, we compute the *total number of predicted collisions* as:

$$\#\texttt{colls}(\mathcal{M}, D) = |\{d \in D \mid \texttt{isCollision}(\mathcal{M}, d)\}| \tag{4}$$

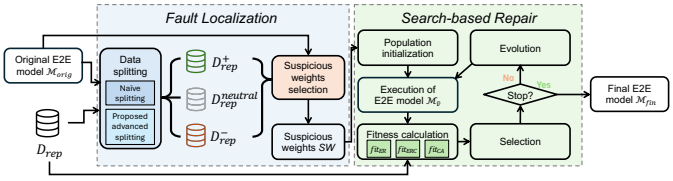



Fig. 1: Overview of E2ERep

## III. E2ERep: Proposed E2E model repair approach

In this section, we propose *E2E repair* (E2ERep), a repair approach for trajectory planning in E2E ADSs. The overview of the approach is shown in Fig. 1. The approach takes as input an E2E model $\mathcal{M}_{orig}$, and a *repair dataset* $D_{rep}$ of inputs for the model. For some of these inputs, the model produces safe planning trajectories, while, for some others, it produces unsafe trajectories that can lead the vehicle to collision or close to it. E2ERep operates in two steps. In the *fault localization* step (see §III-A), the approach identifies some *suspicious weights* $SW$ of $\mathcal{M}_{orig}$ that could be responsible for the wrong planning decisions. Then, in the *search-based repair* step, it uses a search algorithm to find a better setting of the weights $SW$ that leads to a better performance over $D_{rep}$. We build E2ERep on Arachne [6], a widely used DNN repair approach that first localizes suspicious weights responsible for DNN misbehavior and then directly repairs these weights through search-based optimization. We acknowledge that other DNN repair approaches are also relevant; their applicability in repairing E2E models is discussed in §VIII.

### A. Fault localization

In this step, the approach identifies the weights of the E2E model that are responsible for wrong planning decisions over the repair data $D_{rep}$. The approach first splits the repair data as described in §III-A1, and then selects the suspicious weights of the model as described in §III-A2.

*1) Data splitting:* To localize faulty weights, the approach must assess the influence of the weights on the inferences for inputs in which the model behaves correctly (*positive inputs* $D^{+}_{rep}$), and for the inputs on which it does not behave correctly (*negative inputs* $D^{-}_{rep}$). However, E2E models are regression models, for which there is no such clear splitting of the inputs. Indeed, the performance of the model is assessed with quantitative metrics like L2 error, and setting a hard threshold, below which the performance of the model should be considered correct, is not easy. In the following, we describe two approaches to perform such splitting: a naive one taken from the literature, and our newly proposed one.

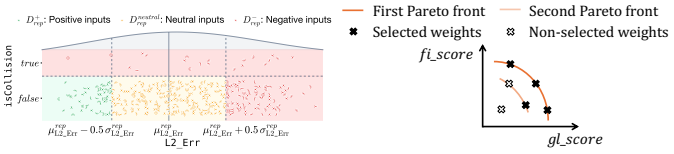



Fig. 2: Visualization of data splitting $adv$




Fig. 3: Ex: Selection of 4 weights from the Pareto fronts

*a) Naive splitting ($naive$):* In the literature, the problem of fault localization for regression models (specifically for semantic segmentation models) has been considered by Schneider et al. [13]. In that work, the authors sort the repair data by the error metric (`mIoU` in their case), and then use the median value as separator, so obtaining the two sets of positive and negative inputs (i.e., $D^+_{rep}$ and $D^-_{rep}$) of equal sizes. However, that approach completely disregards the real distribution of the error metric: in cases in which most of the inputs have high error, still half of them would be considered positive, while most of them should be considered negative; similarly, in cases in which most of the inputs have very low error values, half of them would still be considered negative, while most of them should be considered positive.

*b) Proposed advanced splitting ($adv$):* To address the issues of the $naive$ splitting method, in this paper we propose a more advanced splitting method (called $adv$) tailored for E2E models. The approach considers both L2 error and collision metrics. Given the input model $\mathcal{M}_{orig}$ and the repair data $D_{rep}$, the approach computes the mean L2 error $\mu^{rep}_{\texttt{L2\_Err}}$ (see Eq. 2) and standard deviation $\sigma^{rep}_{\texttt{L2\_Err}}$ of the `L2_Err` of the inferences of $\mathcal{M}_{orig}$ over $D_{rep}$. Then, it classifies each input $d \in D_{rep}$ by applying these rules in order:

(1) if $\texttt{isCollision}(\mathcal{M}_{orig}, d)$ is *true* or $\texttt{L2\_Err}(\mathcal{M}_{orig}, d)$ is greater than $(\mu^{rep}_{\texttt{L2\_Err}} + 0.5 \times \sigma^{rep}_{\texttt{L2\_Err}})$, $d$ is added to $D^-_{rep}$ (i.e., the set of negative inputs); the intuition is that an input is considered *negative* if the planned trajectory leads to a collision or the L2 error is big enough (i.e., significantly more than the mean) so that it indicates large deviations from the optimal trajectory;

(2) if $\texttt{L2\_Err}(\mathcal{M}_{orig}, d)$ is lower than $(\mu^{rep}_{\texttt{L2\_Err}} - 0.5 \times \sigma^{rep}_{\texttt{L2\_Err}})$, $d$ is added to $D^+_{rep}$; the intuition is that the model produces an almost optimal trajectory in this case, and so it is considered *positive*;

(3) in the remaining cases, $d$ is added to the set of inputs $D^{neutral}_{rep}$ that are *neutral*, i.e., neither good nor bad. The intuition is that these inputs do not expose clear deficiencies of the model but, also, they are not clear cases in which the model takes good decisions. Therefore, we discard these inputs from fault localization, as considering them may negatively affect the selection of faulty weights.

A visualization of the splitting process is reported in Fig. 2. The figure combines two plots: a scatter plot showing, for each point, the `L2_Err` and whether there is a collision or not; and a marginal density plot showing their distribution w.r.t. `L2_Err`.

*2) Suspicious weights selection:* Given the positive and negative repair datasets $D^+_{rep}$ and $D^-_{rep}$, the approach selects a set of *suspicious weights*, i.e., weights that are more likely responsible for the wrong planned trajectories of the model.

By following the classical DNN repair approach [6], for each weight $w \in \mathcal{M}_{orig}.W$, it computes:

- the *forward impact* of $w$ indicating the contribution of the weight to the final output. We indicate with $fi^+$ the total forward impact over positive data $D^+_{rep}$, and with $fi^-$ the total forward impact over negative data $D^-_{rep}$;
- the *gradient loss* of $w$ indicating the contribution of the weight to the final error. Similarly as before, we use $gl^+$ and $gl^-$ to indicate the absolute gradient loss over positive and negative repair data;
- finally, it computes the following metrics:

$$fi\_score_w = \frac{fi^-}{1+fi^+} \qquad gl\_score_w = \frac{gl^-}{1+gl^+} \tag{5}$$

The intuition is that the higher $fi\_score_w$, the more weight $w$ is related to negative inputs, and, the higher $gl\_score_w$, the more $w$ is related to wrong inferences. Constant 1 is to avoid division by zero.

Finally, we add to the set of *suspicious weights* $SW$, the top percentage $wr$ of weights in $\mathcal{M}_{orig}.W$ that maximize both metrics, as these are the most influential weights that are also more correlated with the errors (thus, we select $wr \times |\mathcal{M}_{orig}.W|$ weights). Specifically, we arrange the weights in consecutive Pareto fronts w.r.t. the two metrics, and we keep on selecting them from the top front until the desired number of weights has been selected; if not all the weights of the last Pareto front must be selected, we randomly select them. Fig. 3 shows an example of the selection. Assuming that we need to select four weights, we will select all the three weights of the first Pareto front and one weight from the second Pareto front.

### B. Search-based Repair

Given the suspicious weights $SW$ selected in the fault localization phase, we want to find an alternative setting for them such that the performance of the E2E model over $D_{rep}$ improves. We cast this as a single-objective search problem, described in the following.

*1) Search individual:* The *search variables* $\overline{x}$ identify the suspicious weights $SW$, i.e., $\overline{x} = [x_1, \ldots, x_n]$, with $n = |SW|$. An *individual* $\overline{v}$ is a concrete assignment to the search variables, i.e., an alternative setting for the suspicious weights.

The *search space* of $\overline{x}$ is derived from the range of weight values of the original model $\mathcal{M}_{orig}$. Given a variable $x_i$ for the $i$-th suspicious weight $sw_i$, we compute the minimum and maximum values of the weights in the layer of $sw_i$, i.e., $v_{min}$ and $v_{max}$. The search bounds for $x_i$ are then constructed by symmetrically extending this range by 50% on both sides, i.e., $[v_{min} - 0.5 \times \Delta, v_{max} + 0.5 \times \Delta]$, where $\Delta = v_{max} - v_{min}$. This choice should allow to sufficiently explore repair opportunities, without deviating too much from the original model (so avoiding big regression on correct behaviors).

*2) Initial Population:* In the initial population, each individual is initialized by applying a uniformly random perturbation around the original value of each selected suspicious weight,

within $\pm 10\%$ of the weight range $\Delta$ of the corresponding layer (computed as described in §III-B1).

*3) Fitness function:* In a search approach, a *fitness function* is used to select individuals that optimize the given objective. In our case, we want to improve the performance over the repair dataset $D_{rep}$. To do so, we proceed as follows.

Given an individual $\overline{v}$, we replace the values of the weights $SW$ in $\mathcal{M}_{orig}$ with values $\overline{v}$, obtaining a new model $\mathcal{M}_{\overline{v}}$. We then run $\mathcal{M}_{\overline{v}}$ over the repair dataset $D_{rep}$ to assess its quality.

We propose three alternative fitness functions to do so, that differ in the type of information they use. All the three fitness functions need to be minimized.

*a) Error-based (ER):* The first proposed fitness function only uses the cumulative L2 error (see Eq. 1) as guidance to fix wrong behaviors, i.e.,

$$fit_{ER}(\overline{v}) = \sum_{d \in D_{rep}} \texttt{L2\_Err}(\mathcal{M}_{\overline{v}}, d) \quad (6)$$

Minimizing `L2_Err` corresponds to obtaining planned trajectories $\mathbf{o}$ closer to the ground truth trajectories $\hat{\mathbf{o}}$.

*b) Error-collision-based (ERC):* The second proposed fitness starts from the observation that the inputs in which the planned trajectories could lead to collisions, are particularly critical and should be fixed first during the search (irrespective of their L2 error). To do this, we employ a *lexicographic method* [14] in which we consider both the cumulative L2 error (as in fitness $ER$) and the number of collisions as follows:

$$fit_{ERC}(\overline{v}) = \lambda \times \#\texttt{colls}(\mathcal{M}_{\overline{v}}, D_{rep}) + \sum_{d \in D_{rep}} \texttt{L2\_Err}(\mathcal{M}_{\overline{v}}, d) \quad (7)$$

The idea of a lexicographic method is that a component of the fitness has much more importance than another component. In our case, we want that any reduction in the number of collisions (first addend) is more relevant than any reduction in the cumulative L2 error (second addend). To guarantee this, we will set $\lambda$ in an appropriate way, as described in §IV-E.

*c) Category-based (CA):* The previous two fitness functions make use of the cumulative L2 error to (completely or partially) guide the search. One possible issue of that type of guidance is that it tries to reduce the L2 error also for some inputs for which the planned trajectory is good (i.e., not too far from the ground truth trajectory). In this fitness function, we adopt a different strategy that only considers the category of the input, i.e., positive, negative, or neutral as described in §III-A1b. Specifically, we re-evaluate the repair data $D_{rep}$ using model $\mathcal{M}_{\overline{v}}$, and compute the following values:

- $N^{\overline{v}}_{\text{pos}}$ is the number of inputs that are positive;
- $N^{\overline{v}}_{\text{neg,nocol}}$ is the number of inputs that are negative, but do not lead to collisions;
- $N^{\overline{v}}_{\text{neu}}$ is the number of neutral inputs.

The fitness function is defined as follows:

$$fit_{CA}(\overline{v}) = \begin{array}{l} \lambda \times \#\texttt{colls}(\mathcal{M}_{\overline{v}}, D_{rep}) + \\ \alpha \times N^{\overline{v}}_{\text{neg,nocol}} + \beta \times N^{\overline{v}}_{\text{neu}} - \gamma \times N^{\overline{v}}_{\text{pos}} \end{array} \quad (8)$$

Similarly to fitness $ERC$, the first component considers the number of collisions and it is given the highest weight in the function through coefficient $\lambda$, as these are the most critical cases for which the model must be repaired. The second and the third components, instead, consider less critical cases: the negative inputs for which there is no collision, and the neutral inputs. Their coefficients $\alpha$ and $\beta$ are used to define their importance in the fitness. Finally, the last component considers the positive inputs; note that, since the fitness is minimized, the number of positive inputs has a negative sign. Similarly as before, coefficient $\gamma$ defines the importance of this component.

*4) Search Algorithm:* As underlying search algorithm, we adopt Differential Evolution (DE), as it has been successfully applied in other DNN repair approaches for classification problems [6]. The search stops when the total number of generations $MaxGens$ is reached, or there has been no improvement for $H$ generations. As output, the approach returns the model $\mathcal{M}_{fin}$ with the best (i.e., lowest) fitness value.

## IV. Experiment Design

We here present the design of the experiments we conducted to assess E2ERep. Code and all results are available at https://github.com/lyudeyun/E2ERep.

### A. Research Questions (RQs)

**RQ1**: *Does* E2ERep *also improve closed-loop performance at the system level?*
In this RQ, we investigate whether repairing the E2E model using open-loop metrics (as done by E2ERep) leads to improvements in closed-loop metrics.

**RQ2**: *What is the best fault localization approach in* E2ERep*?*
In this RQ, we investigate which of the two fault localization approaches (see §III-A) is more accurate and, therefore, leads to better repair results.

**RQ3**: *What is the best fitness function in* E2ERep*?*
In this RQ, we investigate which of the three fitness functions (see §III-B) leads to better repair results.

**RQ4**: *What is the best weight selection in* E2ERep*?*
In this RQ, we investigate which percentage of selected weights $wr$ (see §III-A) leads to better repair results.

**RQ5**: *What is the most relevant configuration factor in* E2ERep*?*
In this RQ, we investigate which of the three configuration factors affects the repair results the most.

### B. Benchmarks

In the experiments, we considered two representative transformer-based E2E models.

UniAD [3] is a planning-oriented E2E autonomous driving framework that jointly models perception, prediction, and planning within a unified transformer architecture. By directly generating future trajectories from multimodal driving inputs, UniAD enables coordinated decision-making across driving tasks. We took the trained UniAD model available in [12].

VAD [4] is an E2E autonomous driving framework that represents the driving scene using a fully vectorized formulation of map elements and agent dynamics. This representation provides structured constraints for trajectory planning

while improving computational efficiency compared to raster-based approaches. Similarly to UNIAD, VAD directly outputs planned trajectories that can be executed by the downstream controllers. We took the trained VAD model available in [12].

As mentioned in §III, we restrict the repair to the trajectory prediction head, since it directly determines the planned trajectories. Specifically, E2EREP repairs the weights of the fully-connected layers in the trajectory prediction head rather than the weights of the final output layers. In UNIAD, this module contains a single fully-connected layer with 256 $\times$ 256 weights, for a total of 65,536 weights; in VAD it consists of two 512 $\times$ 512 layers, amounting to 524,288 weights.

### C. Datasets

To produce repair and test data, we partition the open-loop validation dataset of `Bench2Drive` [12], which contains 50 clips (i.e., sequences of multiple input frames), into a repair dataset $D_{rep}$ and a test dataset $D_{test}$, with 25 clips each. After removing inputs with incomplete ground truth trajectories, for UNIAD, we obtain 4,935 inputs for $D_{rep}$ and 6,371 for $D_{test}$; for VAD, we obtain 4,685 inputs for $D_{rep}$ and 6,121 for $D_{test}$.

For closed-loop evaluation, we employ two closed-loop evaluation sets from `Bench2Drive` [12], consisting of routes across interactive scenarios in the ADS simulator CARLA [15]. Specifically, `Bench2Drive220`, which consists of 220 routes across 44 interactive scenarios; and `Dev10` [16] which contains ten representative and challenging routes sampled from `Bench2Drive220`.

### D. Compared versions of E2EREP

As explained in §III-A1, E2EREP can use two splitting methods in fault localization, *naive* from [13] and our newly proposed approach *adv*; we identify with $FL_{naive}$ and $FL_{adv}$ the two corresponding fault localization approaches. Moreover, as explained in §III-B3, E2EREP can use three alternative fitness functions ($fit_{ER}$, $fit_{ERC}$, $fit_{CA}$) to guide the search-based repair. So, in total, there are six versions of the approach.

### E. Setting of E2EREP

The fault localization approach requires to select a value $wr$ that identifies the percentage of weights that rank at the top in terms of suspiciousness score and that are selected as suspicious weights $SW$ (see §III-A2). Since UNIAD and VAD differ substantially in the number of weights, applying the same $wr$ would lead to search problems of vastly different complexity. To keep the search space manageable and the computational cost of the repair feasible, for UNIAD, we experiment with $wr \in \{0.01\%, 0.02\%, 0.04\%\}$. For VAD, since it is a larger model (see §IV-B), we use lower percentages; namely, we experiment with $wr \in \{0.005\%, 0.01\%, 0.02\%\}$. We identify the three values as *Low*, *Med*, and *High*.

As explained in §III-B3b, fitness $fit_{ERC}$ adopts a lexicographic method in which the component related to collisions has priority over the component related to cumulative `L2_Err`. To achieve this, we set the coefficient $\lambda$ to 10 so that the solutions with fewer `#colls` are always preferred; this design prevents the search from trading safety with small `L2_Err` improvements. `L2_Err` improvements are only considered for individuals with the same number of collisions.

For fitness $fit_{CA}$ (see §III-B3c), the coefficient $\lambda$ is set to 10, for the same lexicographic strategy of $fit_{ERC}$. Instead, coefficients $\alpha$, $\beta$, and $\gamma$ are set to 1, 0.5, and 1. The intuition is that negative inputs (i.e., inputs with a high `L2_Err`) will receive a moderate penalty, as they indicate inaccurate trajectory predictions. The neutral inputs are slightly penalized to prevent the repair from converging to suboptimal solutions. Finally, positive inputs (i.e., those with a small `L2_Err`) are rewarded to promote correct system behaviors.

As search algorithm (see §III-B), we selected Differential Evolution (DE), a popular algorithm used in other DNN repair approaches [6]. The number of variables $\#Vars$ corresponds to the number of suspicious weights, i.e., $\#Vars = |SW|$. The population size $PopSize$ depends on the number of variables as $PopSize = 2 \times \#Vars$; the intuition is that having more variables requires to have more exploration. As termination condition, we set the maximum number of generations $MaxGens$ to 50 across all experiments; we further stop the search if there is no improvement for $H = 5$ generations.

### F. Running of the experiments

An *experiment* consists of the execution of one approach (one of the six versions of E2EREP; see §IV-D), using a specific percentage of selected weights $wr$ in fault localization (see §IV-E), over one of the two E2E models (see §IV-B). So, in total, there are $6 \times 3 \times 2 = 36$ experiments.

As the repair search algorithm includes some randomness, each experiment has been run 10 times, as suggested by a guideline on experiments with randomized algorithms [17].

### G. Evaluation metrics

For each repetition of each experiment, we compute the mean L2 error (see Eq. 2) and the total number of collisions (see Eq. 4) of the final repaired model $\mathcal{M}_{fin}$ using the test dataset $D_{test}$, i.e., $\mu_{\texttt{L2\_Err}}(\mathcal{M}_{fin}, D_{test})$ and $\#\texttt{colls}(\mathcal{M}_{fin}, D_{test})$. For conciseness, we will call them $\mu_{\texttt{L2\_Err}}^{test,fin}$ and $\#\texttt{colls}^{test,fin}$ in the following.

For answering the five RQs, we proceed as follows. First, for each experiment and each metric `EM` (either $\mu_{\texttt{L2\_Err}}^{test,fin}$ or $\#\texttt{colls}^{test,fin}$), we compute the average $\texttt{EM}_{\texttt{avg}}$ of `EM` across the 10 repetitions of the experiment.

To answer RQ1, we select the best setting of E2EREP (as identified in RQ2, RQ3, and RQ4) for UNIAD and VAD. Then, for each E2E architecture, we run the ten repaired models over the `Dev10` dataset [16], a dataset curated by the `Bench2Drive` developers to guarantee a representative and fast evaluation of models. We evaluate the performance of these ten repaired models using these closed-loop metrics:

- *Driving Score* (*DS*): it combines a *route completion* metric with severity-based penalties for *infractions* (e.g., collisions, traffic rule violations), averaged over all routes.

- *Success Rate* ($SR$): the ratio of routes successfully completed within the time limit without any traffic violations.
- *Efficiency* ($Eff$): the average ratio between the autonomous vehicle speed and the speed of nearby vehicles.
- *Comfortness* ($Comf$): a smoothness-based metric that evaluates whether the autonomous vehicle's motion (e.g., acceleration) remains within predefined bounds derived from human driving behavior.

We compute the Spearman's rank correlation $r_s$ between each open-loop metric (i.e., $\mu_{\texttt{L2\_Err}}$ and $\#\texttt{colls}$) and each closed-loop metric (i.e., $DS$, $SR$, $Eff$, and $Comf$) collected from the ten repaired models.[1] The coefficient $r_s$ ranges in $[-1, 1]$, where the sign indicates *positive* or *negative* correlation, while the magnitude reflects the strength of the correlation. We interpret $r_s$ as follows [18]: *very weak* if $|r_s| \in [0, 0.2)$, *weak* if $|r_s| \in [0.2, 0.4)$, *moderate* if $|r_s| \in [0.4, 0.6)$, *strong* if $r_s \in [0.6, 0.8)$, and *very strong* if $|r_s| \in [0.8, 1.0]$.

Finally, we take the best model and we evaluate it on the full `Bench2Drive220` dataset to have more conclusive results.

To answer RQ2, for each metric `EM`, we perform a pairwise comparison of $\texttt{EM}_{\texttt{avg}}$ across the experiments, between the versions of E2EREP that use $FL_{naive}$ as fault localization approach and those that use $FL_{adv}$. Specifically, we use the non-parametric test Wilcoxon signed-rank test to compare the two distributions, using $\alpha = 0.05$ as the significance level.[2] If the $p$-value is less than $\alpha$, we reject the null hypothesis that there is no significant difference. In case of a significant difference, we use Cohen's $d$ effect size to assess the strength of the significance. If $d > 0$, it means that $FL_{naive}$ leads to higher values of `EM` and, so, $FL_{adv}$ is better (as for both $\mu_{\texttt{L2\_Err}}^{test,fin}$ and $\#\texttt{colls}^{test,fin}$, the lower the better); otherwise, $FL_{naive}$ is better. We further interpret the magnitude of the effect size using the categories from [19]: *small* if $|d| \in (0, 0.2)$, *medium* if $|d| \in [0.2, 0.8)$, and *large* if $|d| \geq 0.8$.

To answer RQ3, we proceed similarly to RQ2, by performing, for each metric `EM`, a pairwise comparison of $\texttt{EM}_{\texttt{avg}}$ across the experiments, among the versions of E2EREP that use the three fitness functions ($fit_{ER}$, $fit_{ERC}$, $fit_{CA}$).

To answer RQ4, we proceed similarly to RQ2 and RQ3, by doing, for each metric `EM`, a pairwise comparison of $\texttt{EM}_{\texttt{avg}}$ across the experiments, among the E2EREP versions that use the three weight selection percentages $wr$ ($Low$, $Med$, $High$).

To answer RQ5, we assess the sensitivity of E2EREP to its configuration factors using a $2 \times 3 \times 3$ factorial design over fault localization approach $FL$, fitness function $fit$, and weight selection ratio $wr$ (10 runs per configuration). For each metric `EM`, we use the aligned rank transform (ART) to conduct a non-parametric multi-factor ANOVA. We consider both the main effect of each configuration factor and the two-way and three-way interaction effects among them. We report $p$-values for statistical significance and partial eta squared ($\eta_p^2$) for effect size. Effects with $p < 0.05$ are considered statistically significant. Following Cohen's ANOVA effect-size classification [20], we interpret $\eta_p^2$ values in $[0, 0.01)$ as negligible, $[0.01, 0.06)$ as *small*, $[0.06, 0.14)$ as *medium*, and $[0.14, 1]$ as *large*.

[1] We use a non-parametric test because the Shapiro–Wilk test indicated that the distributions are not normal.

[2] As before, we use a non-parametric test as the distributions are not normal.

TABLE I: RQ1 – Results of the original E2E model $\mathcal{M}_{orig}$ and the 10 models repaired with the best setting of E2EREP, using open-loop and closed loop metrics over `Dev10` dataset

(a) UNIAD

| | Open-loop ↓ | | Closed-loop ↑ | | | |
|---|---|---|---|---|---|---|
| | $\mu_{\texttt{L2\_Err}}$ | $\#\texttt{colls}$ | $DS$ | $SR$ (%) | $Eff$ | $Comf$ |
| $\mathcal{M}_{orig}$ | 1.2513 | 588 | 51.41 | 20 | 95.65 | 36.74 |
| $\mathcal{M}_{fin}^{1}$ | 1.2519 | 592 | 39.38 | 10 | 95.04 | 34.92 |
| $\mathcal{M}_{fin}^{2}$ | 1.2603 | **509** | 49.14 | **30** | 99.35 | 36.65 |
| $\mathcal{M}_{fin}^{3}$ | 1.2498 | 569 | 41.52 | 10 | 96.43 | 37.88 |
| $\mathcal{M}_{fin}^{4}$ | 1.2518 | 584 | 39.75 | 10 | 95.46 | 39.40 |
| $\mathcal{M}_{fin}^{5}$ | 1.2510 | 555 | 40.37 | 20 | 91.55 | 47.94 |
| $\mathcal{M}_{fin}^{6}$ | 1.2504 | 574 | 50.35 | **30** | 93.70 | 36.00 |
| $\mathcal{M}_{fin}^{7}$ | **1.2486** | 516 | 40.04 | 20 | 93.24 | 41.26 |
| $\mathcal{M}_{fin}^{8}$ | 1.2504 | 545 | 49.26 | 20 | **99.71** | 35.94 |
| $\mathcal{M}_{fin}^{9}$ | 1.2490 | 580 | **52.38** | **30** | 95.02 | 35.53 |
| $\mathcal{M}_{fin}^{10}$ | 1.2491 | 565 | 41.51 | 20 | 88.05 | 42.64 |

(b) VAD

| | Open-loop ↓ | | Closed-loop ↑ | | | |
|---|---|---|---|---|---|---|
| | $\mu_{\texttt{L2\_Err}}$ | $\#\texttt{colls}$ | $DS$ | $SR$ (%) | $Eff$ | $Comf$ |
| $\mathcal{M}_{orig}$ | 1.4263 | 80 | 41.86 | **20** | 95.76 | 36.32 |
| $\mathcal{M}_{fin}^{1}$ | 1.4066 | 71 | **48.31** | **20** | **105.82** | 37.04 |
| $\mathcal{M}_{fin}^{2}$ | 1.4052 | **66** | 27.70 | **20** | 95.01 | 40.53 |
| $\mathcal{M}_{fin}^{3}$ | 1.3890 | 76 | 39.45 | **20** | 95.29 | 43.7 |
| $\mathcal{M}_{fin}^{4}$ | 1.3974 | 71 | 38.33 | 0 | 104.70 | 39.89 |
| $\mathcal{M}_{fin}^{5}$ | 1.3888 | 77 | 43.19 | **20** | 90.65 | 41.15 |
| $\mathcal{M}_{fin}^{6}$ | 1.3871 | 77 | 39.14 | **20** | 93.86 | 37.97 |
| $\mathcal{M}_{fin}^{7}$ | 1.3953 | 68 | 36.27 | 10 | 105.71 | 40.89 |
| $\mathcal{M}_{fin}^{8}$ | 1.4077 | **66** | 43.97 | **20** | 86.99 | 42.91 |
| $\mathcal{M}_{fin}^{9}$ | **1.3870** | 70 | 26.42 | 0 | 98.75 | **45.79** |
| $\mathcal{M}_{fin}^{10}$ | 1.3912 | 78 | 33.99 | 0 | 96.92 | 44.9 |

### H. Experiments platform

Experiments were conducted on a server with an Intel Core i9-14900K CPU, 128 GB RAM, and an NVIDIA RTX PRO 6000 Blackwell Max-Q Workstation Edition GPU.

## V. EXPERIMENTAL RESULTS

We here answer the different RQs (see §IV-A). Boxplots with the distributions across 10 runs of $\mu_{\texttt{L2\_Err}}^{test,fin}$ and $\#\texttt{colls}^{test,fin}$ for the different versions of E2EREP are reported in the online repository.

*RQ1: Impact of open-loop repair on closed-loop performance*

As explained in §I, the optimal way to assess an E2E model is by running it in a closed-loop simulation (over a set of scenarios) to observe how it affects the driving safety and efficiency. However, such evaluation is too costly and cannot be used during repair. Therefore, in E2EREP, we investigated the use of open-loop metrics (see fitness functions in §III-B3) that only check the performance of single inferences, without running expensive simulations. In this RQ, we assess whether this is effective, i.e., if improving open-loop metrics can lead to better closed-loop performance at the system level. Since assessing closed-loop performance is very costly (around 30 mins per scenario), we proceed as follows. For both UNIAD and VAD, we select the best version of E2EREP identified in RQ2, RQ3, and RQ4. Then, we execute the 10 models obtained by running this best version (i.e., in the 10 repetitions of E2EREP) over the closed-loop scenarios of the `Dev10` dataset. Table I reports the results of the original model $\mathcal{M}_{orig}$ and of the 10 repaired models in terms of the open-loop metrics *mean L2 error* ($\mu_{\texttt{L2\_Err}}$) and *number of predicted collisions* ($\#\texttt{colls}$) described in §II-B, and closed-loop metrics *driving score* ($DS$), *success rate* ($SR$), *efficiency* ($Eff$), and *comfortness* ($Comf$), as described in §IV-G.

Since open-loop metrics should be minimized and closed-loop metrics should be maximized, a good result would be to have a negative correlation between the two, i.e., minimizing

TABLE II: RQ1 – Spearman rank correlation between open-loop and closed-loop metrics across the ten repaired models in Table I. (Correlation strength: positive correlation – very weak, weak, moderate, strong, and very strong; negative correlation – very weak, weak, moderate, strong, and very strong.)

| | | Closed-loop Metric ↑ | | | |
|---|---|---|---|---|---|
| | **Open-loop Metric ↓** | *DS* | *SR* | *Eff* | *Comf* |
| UniAD | $\mu_{\texttt{L2\_Err}}$ | -0.29 | -0.16 | 0.44 | -0.19 |
| | $\#\texttt{colls}$ | -0.21 | -0.40 | -0.07 | -0.42 |
| VAD | $\mu_{\texttt{L2\_Err}}$ | 0.39 | 0.29 | 0.10 | -0.38 |
| | $\#\texttt{colls}$ | 0.29 | 0.30 | 0.41 | -0.89 |

TABLE III: RQ1 – Results of the original E2E model $\mathcal{M}_{orig}$ and the best repaired model $\mathcal{M}_{fin}$ on `Bench2Drive220`

| a | | Open-loop Metric ↓ | | Closed-loop Metric ↑ | | | |
|---|---|---|---|---|---|---|---|
| | | $\mu_{\texttt{L2\_Err}}$ | $\#\texttt{colls}$ | *DS* | *SR* (%) | *Eff* | *Comf* |
| UniAD | $\mathcal{M}_{orig}$ | 1.2513 | 588 | 43.84 | 13.64 | 130.10 | 38.35 |
| | $\mathcal{M}^{9}_{fin}$ | 1.2490 | 580 | 43.43 | 13.18 | 130.38 | 37.77 |
| VAD | $\mathcal{M}_{orig}$ | 1.4263 | 80 | 43.87 | 15.00 | 157.26 | 40.47 |
| | $\mathcal{M}^{1}_{fin}$ | 1.4066 | 71 | 38.53 | 13.18 | 150.26 | 46.56 |

the open-loop metrics leads to an improvement of closed-loop metrics. To check if this is the case, we perform correlation analysis between the open-loop metrics and the closed-loop metrics using the Spearman correlation test; Table II reports the results. For UniAD, both open-loop metrics inversely correlate with almost all closed-loop metrics (except for $\mu_{\texttt{L2\_Err}}$ with *Eff*), with medium strength in two cases. Such results are good, as they show that E2ERep can lead to better closed-loop performance in terms of safety and compliance to traffic rules; the results on *Eff* probably mean that safety is obtained by driving more carefully (i.e., lower speed).

However, the results for VAD are less good. For both $\mu_{\texttt{L2\_Err}}$ and $\#\texttt{colls}$, there is a positive correlation with three closed-loop metrics that are related to the safety and effectiveness of driving (*DS*, *SR*, *Eff*), indicating that E2ERep does not allow to improve the driving in closed-loop evaluation.

To have more definitive conclusions, for each E2E architecture, we take the best repaired model from Table I ($\mathcal{M}^{9}_{fin}$ for UniAD and $\mathcal{M}^{1}_{fin}$ for VAD) and evaluate it on the much larger dataset `Bench2Drive220`, composed of 220 driving scenarios. Results of open-loop and closed-loop metrics are reported in Table III. We observe that, for UniAD, closed-loop metrics in $\mathcal{M}_{fin}$ remain almost unchanged w.r.t. $\mathcal{M}_{orig}$, with small degradations or small improvements. For VAD, instead, *DS*, *SR*, and *Eff* decrease, while *Comf* improves. This shows that, overall, repair based on open-loop metrics does not guarantee to improve the closed-loop performance.

**Answer to RQ1**: Overall, we observe that using open-loop metrics in E2E repair (as done by E2ERep) does not necessarily improve closed-loop metrics and, in some cases, may even decrease them.

TABLE IV: RQ2 – Statistical comparison between $FL_{naive}$ and $FL_{adv}$. (≡: no difference between the two approaches. ✓, ✓✓, ✓✓✓: the approach on the row is significantly better than the approach on the column with strength *small*, *medium*, *large*. ✗, ✗✗, ✗✗✗: the approach on the row is significantly worse than the approach on the column with strength *small*, *medium*, *large*). The best approach is highlighted in green.

(a) UniAD

| | $\mu^{test,fin}_{\texttt{L2\_Err}}$ | | $\#\texttt{colls}^{test,fin}$ | |
|---|---|---|---|---|
| | $FL_{naive}$ | $FL_{adv}$ | $FL_{naive}$ | $FL_{adv}$ |
| $FL_{naive}$ | ≡ | ✗ | ≡ | ✗✗✗ |
| $FL_{adv}$ | ✓ | ≡ | ✓✓✓ | ≡ |

(b) VAD

| | $\mu^{test,fin}_{\texttt{L2\_Err}}$ | | $\#\texttt{colls}^{test,fin}$ | |
|---|---|---|---|---|
| | $FL_{naive}$ | $FL_{adv}$ | $FL_{naive}$ | $FL_{adv}$ |
| $FL_{naive}$ | ≡ | ✗✗✗ | ≡ | ✗✗ |
| $FL_{adv}$ | ✓✓✓ | ≡ | ✓✓ | ≡ |

TABLE V: RQ3 – Statistical comparison between the fitness functions $fit_{ER}$, $fit_{ERC}$, and $fit_{CA}$. Legend as in Table IV

(a) UniAD

| | $\mu^{test,fin}_{\texttt{L2\_Err}}$ | | | $\#\texttt{colls}^{test,fin}$ | | |
|---|---|---|---|---|---|---|
| | $fit_{ER}$ | $fit_{ERC}$ | $fit_{CA}$ | $fit_{ER}$ | $fit_{ERC}$ | $fit_{CA}$ |
| $fit_{ER}$ | ≡ | ✓✓ | ✓✓✓ | ≡ | ✗✗✗ | ✗✗✗ |
| $fit_{ERC}$ | ✗✗ | ≡ | ✓✓ | ✓✓✓ | ≡ | ✓ |
| $fit_{CA}$ | ✗✗✗ | ✗✗ | ≡ | ✓✓✓ | ✗ | ≡ |

(b) VAD

| | $\mu^{test,fin}_{\texttt{L2\_Err}}$ | | | $\#\texttt{colls}^{test,fin}$ | | |
|---|---|---|---|---|---|---|
| | $fit_{ER}$ | $fit_{ERC}$ | $fit_{CA}$ | $fit_{ER}$ | $fit_{ERC}$ | $fit_{CA}$ |
| $fit_{ER}$ | ≡ | ✓✓ | ✓✓✓ | ≡ | ✗✗✗ | ✗✗✗ |
| $fit_{ERC}$ | ✗✗ | ≡ | ✓✓✓ | ✓✓✓ | ≡ | ✗ |
| $fit_{CA}$ | ✗✗✗ | ✗✗✗ | ≡ | ✓✓✓ | ✓ | ≡ |

*RQ2: Impact of fault localization on repair performance*

In this RQ, we want to observe whether the fault localization approach $FL_{adv}$ is more effective than $FL_{naive}$ at identifying weights whose repair leads to improved system behavior.

The statistical comparison in Table IV shows that $FL_{adv}$ consistently outperforms $FL_{naive}$ across both models and both metrics. For `L2_Err`, $FL_{adv}$ yields a lower `L2_Err`, with effect sizes ranging from *small* for UniAD to *large* for VAD. This indicates that the weights localized by $FL_{adv}$ are more strongly associated with the production of planned trajectories that deviate from the optimal trajectories.

For $\#\texttt{colls}$, the results are even more pronounced. $FL_{adv}$ performs better with a *large* effect size for UniAD and a *medium* effect size for VAD. This indicates that fixing the weights selected by $FL_{adv}$ allows to reduce the number of possible collisions.

**Answer to RQ2**: The newly proposed fault localization method $FL_{adv}$ provides statistically significant better results in terms of $\mu^{test,fin}_{\texttt{L2\_Err}}$ and $\#\texttt{colls}^{test,fin}$, for both UniAD and VAD.

*RQ3: Impact of fitness function on repair performance*

In this RQ, we are interested in assessing which fitness function provides better guidance during repair. Table V reports the statistical comparison of the different fitness functions. Regarding $\mu^{test,fin}_{\texttt{L2\_Err}}$, we observe that $fit_{ER}$ is statistically significantly better than $fit_{ERC}$ and $fit_{CA}$ with, respectively, *medium* and *large* effect sizes in both UniAD and VAD. This is expected, as the only goal of $fit_{ER}$ is to improve (i.e., reduce) the mean `L2_Err` (see Eq. 6). $fit_{ERC}$ is better than $fit_{CA}$ (with *medium* and *large* effect sizes for UniAD and VAD) because, although $fit_{ERC}$ primarily considers the number of collisions, it also has a component related to mean

TABLE VI: RQ4 – Statistical comparison between weight selection percentages (*Low*, *Med*, *High*). Legend as in Table IV

(a) UNIAD

| | $\mu_{\texttt{L2\_Err}}^{test,fin}$ | | | $\#\texttt{colls}^{test,fin}$ | | |
|---|---|---|---|---|---|---|
| | *Low* | *Med* | *High* | *Low* | *Med* | *High* |
| *Low* | ≡ | ✗✗ | ✗ | ≡ | ✗✗✗ | ✗✗ |
| *Med* | ✓✓ | ≡ | ✓ | ✓✓✓ | ≡ | ✓ |
| *High* | ✓ | ✗ | ≡ | ✓✓ | ✗ | ≡ |

(b) VAD

| | $\mu_{\texttt{L2\_Err}}^{test,fin}$ | | | $\#\texttt{colls}^{test,fin}$ | | |
|---|---|---|---|---|---|---|
| | *Low* | *Med* | *High* | *Low* | *Med* | *High* |
| *Low* | ≡ | ✓✓ | ✓✓✓ | ≡ | ✗✗✗ | ✗✗✗ |
| *Med* | ✗✗ | ≡ | ✓✓ | ✓✓✓ | ≡ | ✗✗ |
| *High* | ✗✗✗ | ✗✗ | ≡ | ✓✓✓ | ✓✓ | ≡ |

`L2_Err` (see Eq. 7); $fit_{CA}$, instead, only considers the category of the inputs (see Eq. 8) and, therefore, it is not good at just improving the `L2_Err`.

Regarding $\#\texttt{colls}^{test,fin}$, the results are different. $fit_{ERC}$ and $fit_{CA}$ are both statistically significantly better than $fit_{ER}$ in both models, with *large* effect size. The result is expected, as both fitness functions have a component related to the number of collisions that is more important (i.e., higher coefficient in the definition) than the other components. Regarding the comparison between $fit_{ERC}$ and $fit_{CA}$, the results are mixed; $fit_{ERC}$ is better than $fit_{CA}$ for UNIAD, while $fit_{CA}$ is better for VAD; however, in both cases the effect size is *small*, indicating that, probably, the difference between the two fitness functions is minimal. Indeed, they both share a main component related to collisions that drives the search.

Previous results show that reducing the number of collisions, does not necessarily lead to large reductions in cumulative `L2_Err`, as collisions can be avoided by taking different trajectories that, however, are still very different from the ground truth trajectories (so, still having high `L2_Err`). Similarly, reducing the cumulative `L2_Err` does not necessarily lead to reductions of the number of collisions; indeed, some negative inputs without collision have a high `L2_Err`, and reducing them does not lead to any reduction in the number of collisions.

To conclude, what fitness function to use depends on the metric of interest. If users are more interested in `L2_Err`, they should use $fit_{ER}$, while if they care more about the number of collisions they should opt for $fit_{ERC}$ or $fit_{CA}$.

**Answer to RQ3**: $fit_{ER}$ is consistently the best fitness function for improving $\mu_{\texttt{L2\_Err}}^{test,fin}$, while $fit_{ERC}$ and $fit_{CA}$ are better for $\#\texttt{colls}^{test,fin}$.

*RQ4: Impact of weight selection ratio on repair performance*

In this RQ, we are interested in assessing what is the best percentage of weight selection $wr$ that is used to select the top suspicious weights in fault localization (see §III-A2). As explained in §IV-E, we experiment with three values of $wr$ (*Low*, *Med*, *High*) that select a low, medium, and high number of weights. Selecting more weights gives more opportunities for repair but, at the same time, increases the search space and can slow down the search. On the other hand, repairing fewer weights gives fewer opportunities for repair.

Table VI summarizes the statistical comparison among the three weight selection ratios. For UNIAD (Table VIa), for both metrics, *Med* provides the best results (with *medium* to *large* effect sizes w.r.t. *Low*, and *small* effect sizes w.r.t. *High*), followed by *High*. This suggests that, for UNIAD, repairing too few weights may limit the search space, while further increasing the number of selected weights beyond *Med* does not bring additional benefits.

For VAD (Table VIb), instead, the results are mixed and provide no clear trend with respect to the percentage of selected weights. While *Low* is better for $\mu_{\texttt{L2\_Err}}^{test,fin}$ (with *medium* and *large* effect sizes w.r.t. *Med* and *High*), *High* is better for $\#\texttt{colls}^{test,fin}$ (with *large* and *medium* effect sizes w.r.t. *Low* and *Med*). This suggests that, for VAD, the percentage of weight selection $wr$ involves a trade-off: a larger set of repaired weights may be needed to reduce collision-related failures, whereas a smaller set can be more beneficial for improving trajectory accuracy.

To summarize, all previous results indicate that the selection of the number of weights to repair is critical, and different E2E models and metrics may require different quantities.

**Answer to RQ4**: Overall, the optimal number of weights to repair depends on the model and the metric. In UNIAD, *Med* is the best for both metrics; in VAD, instead, *Low* is better for $\mu_{\texttt{L2\_Err}}^{test,fin}$, and *High* for $\#\texttt{colls}^{test,fin}$. So, a moderate ratio may offer a reasonable trade-off.

TABLE VII: RQ5 – Sensitivity analysis of configuration factors. Significant $p$-values ($p < 0.05$) are shown in **bold**. Effect size $\eta_p^2$: negligible, small, medium, and large.

(a) UNIAD

| | $\mu_{\texttt{L2\_Err}}^{test,fin}$ | | $\#\texttt{colls}^{test,fin}$ | |
|---|---|---|---|---|
| | $p$-value | $\eta_p^2$ | $p$-value | $\eta_p^2$ |
| FL | 0.3858 | 0.0056 | **0.0266** | 0.0361 |
| *fit* | **0.0237** | 0.0543 | **0.0002** | 0.1193 |
| *wr* | 0.6172 | 0.0072 | 0.8448 | 0.0025 |
| FL:*fit* | 0.4778 | 0.0110 | 0.3674 | 0.0148 |
| FL:*wr* | **0.0354** | 0.0486 | **0.0136** | 0.0621 |
| *fit*:*wr* | 0.5610 | 0.0218 | 0.0640 | 0.0637 |
| FL:*fit*:*wr* | 0.3897 | 0.0301 | 0.8616 | 0.0096 |

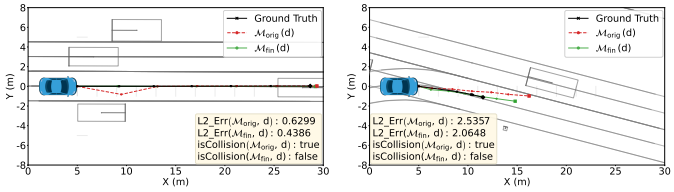
(b) VAD

| | $\mu_{\texttt{L2\_Err}}^{test,fin}$ | | $\#\texttt{colls}^{test,fin}$ | |
|---|---|---|---|---|
| | $p$-value | $\eta_p^2$ | $p$-value | $\eta_p^2$ |
| FL | **0.0000** | 0.1347 | 0.4283 | 0.0039 |
| *fit* | 0.1066 | 0.0271 | 0.3640 | 0.0123 |
| *wr* | 0.5101 | 0.0082 | 0.0557 | 0.0348 |
| FL:*fit* | 0.6855 | 0.0046 | 0.5484 | 0.0073 |
| FL:*wr* | 0.2524 | 0.0167 | 0.6440 | 0.0054 |
| *fit*:*wr* | 0.8626 | 0.0079 | 0.9666 | 0.0035 |
| FL:*fit*:*wr* | 0.9863 | 0.0021 | 0.8111 | 0.0096 |

*RQ5: Most relevant configuration factor in* E2EREP

While RQ2, RQ3, and RQ4 studied the best settings of the three configuration factors of E2EREP, in this RQ we investigate which configuration factor is overall the most relevant to the effectiveness of the approach. We performed a sensitivity analysis of E2EREP w.r.t. the three configuration factors across the two models and two metrics (see §IV-G). Results are reported in Table VII. Overall, the results show that no single factor consistently dominates across both models and both metrics. Instead, the most influential factor depends on the model and metric.

For UNIAD (Table VIIa), *fit* shows the strongest main effect. It significantly affects both $\mu_{\texttt{L2\_Err}}^{test,fin}$ and $\#\texttt{colls}^{test,fin}$, with small and medium effect sizes, respectively. *FL* also significantly affects $\#\texttt{colls}^{test,fin}$, but with a small effect size. In contrast, *wr* does not show a significant effect on either metric. However, the significant *FL*:*wr* interactions for




(a) UNIAD (b) VAD

Fig. 4: Example of improvement on open-loop metrics – Planned trajectories before and after repair

both $\mu_{\texttt{L2\_Err}}^{test,fin}$ and $\#\texttt{colls}^{test,fin}$ suggest that the choice of $wr$ should be considered together with the fault localization strategy, rather than independently.

For VAD (Table VIIb), *FL* is the dominant factor for $\mu_{\texttt{L2\_Err}}^{test,fin}$, with a medium effect size; the other factors are not statistically significant. For $\#\texttt{colls}^{test,fin}$, no factor reaches statistical significance, although $wr$ shows a marginal tendency. None of the two-way or three-way interactions is significant for VAD, suggesting that the tested factors do not exhibit clear interaction effects for VAD.

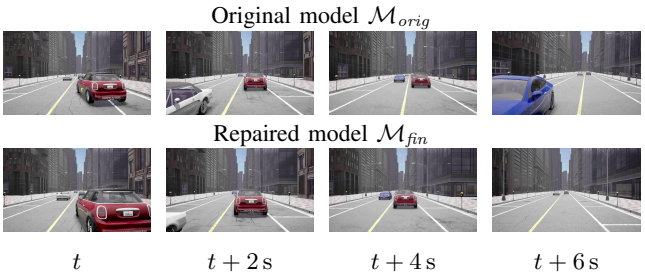
**Answer to RQ5**: The most relevant configuration factor depends on the model and metric. For UNIAD, *fit* is the most influential factor, especially for $\#\texttt{colls}^{test,fin}$. For VAD, *FL* is the most influential factor for $\mu_{\texttt{L2\_Err}}^{test,fin}$. For both E2E architectures, $wr$ shows no significant effect.

## VI. DISCUSSION

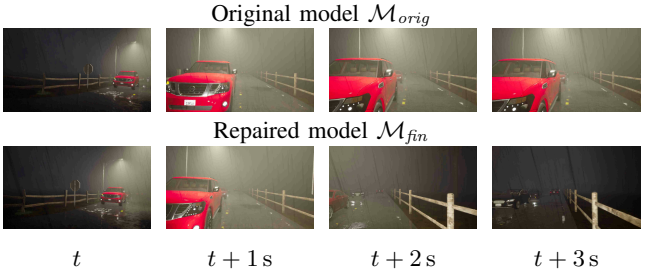
We here provide a more critical discussion. This work has been driven by our collaboration with an automotive company that is investigating the adoption and development of E2E models. In their context, having an approach that can quickly fix an E2E model against emerging failures, is important for fast development and prototyping. To this end, DNN repair seems a promising approach (compared to expensive retraining), given its success for classical DNN models.

Therefore, in this practical experience report, we proposed a repair approach for E2E models. As explained in §I, using closed-loop metrics for the evaluation of candidate models during repair is not feasible and, therefore, we investigated the adoption of open-loop metrics that are faster to compute but, possibly, do not correlate with closed-loop performance.

Results of RQ2, RQ3, and RQ4 show that E2EREP is effective in improving open-loop metrics. To understand how repair works, Fig. 4 shows the effect of repair on two frames of UNIAD and VAD. In the figures, the black line is the ground truth trajectory (i.e., the trajectory that should be planned by an optimal model), the red line is the trajectory planned by the original model $\mathcal{M}_{orig}$, and the green one by the repaired model $\mathcal{M}_{fin}$. For UNIAD (Fig. 4a), the original model plans a trajectory to the right that can lead to a collision with a parked car, while the repaired trajectory aligns with the straight ground truth trajectory. For the VAD model (Fig. 4b), the planned trajectory leads to a possible collision with the vehicle




(a) UNIAD – RouteScenario_18305 of `Bench2Drive220`




(b) VAD – RouteScenario_2397 of `Bench2Drive220`

Fig. 5: Example of improvement on closed-loop metrics – Simulations (frames from left to right) of the original model $\mathcal{M}_{orig}$ (top row) and the repaired model $\mathcal{M}_{fin}$ (bottom row). The images show the point of view of the autonomous vehicle.

to the left, while the repaired trajectory is more aligned with the ground truth trajectory and avoids the collision.

However, the final goal of the repair is to improve the performance of the E2E model in the whole driving scenario (measured by closed-loop metrics). Fig. 5 shows, for the two E2E architectures, two scenarios in which the original model $\mathcal{M}_{orig}$ leads to a collision (top row), while the repaired model avoids the collision (bottom row). In the original model of UNIAD (top row in Fig. 5a), a red vehicle cuts in front of the autonomous vehicle (first frame), that decides to go on the left lane to avoid hitting the red vehicle, but it ends up colliding with the blue vehicle coming in the opposite direction (last frame). In the repaired model (bottom row), instead, the autonomous vehicle gets closer to the red vehicle and then decides to decelerate to safely avoid the collision.

In the original model of VAD (top row of Fig. 5b), the autonomous vehicle takes too large a right turn and collides with the incoming red vehicle. In the repaired model (bottom row), instead, the autonomous vehicle drives slower and makes a sharper turn, so avoiding the collision.

Despite these good cases, in RQ1, we observed that, overall, improving open-loop metrics does not always guarantee to improve closed-loop metrics. This calls for more research in this area. We identified these possible research directions:

- *Selection of the repair data $D_{rep}$*: in E2EREP, we selected $D_{rep}$ based on open-loop metrics (some with low `L2_Err` and $\#\texttt{colls}$, and some with high). However, since these do not necessarily reflect closed-loop perfor-

mance, a possible research direction could be to select the frames to repair from the closed-loop simulations, e.g., the frames close to a collision. This would only increase a little bit the cost of data selection, as it would require performing some closed-loop simulations (but only once).
- *Use of a limited number of closed-loop simulations*: another possible solution could be to run, during the search, the expensive closed-loop simulations only on a restricted number of individuals (e.g., the best two or three at each generation), and adjust the fitness value based on closed-loop metrics. This would have the advantage of realigning the search with the closed-loop performance, and of avoiding assigning good fitness values to individuals that are good in terms of open-loop metrics but they are actually not so good in terms of closed-loop performance, and the other way round. Of course, this would incur a higher computational cost, but still lower than executing the closed-loop simulation for all individuals.
- *Low-fidelity simulator*: another solution could be to perform closed-loop simulation in fitness evaluations, using fast low-fidelity simulators (like CommonRoad) that abstract some aspects of driving like perception. However, doing so is non-trivial, as it requires building bridges between the E2E model and the simulator. Moreover, it is not clear how effective these abstractions can be. Another possible solution could be building surrogate models mimicking ADS simulation, but also building the surrogate model has a non-negligible cost.

## VII. Threats to validity

*Construct validity.* In our evaluation, we mainly adopt L2 error and the number of collisions as our metrics to assess the repaired E2E models. As introduced in §II-B, these metrics are commonly adopted to evaluate E2E trajectory planning models, and we use them across different variations of our approach to ensure the consistency of the results. Moreover, we assume that the architecture of the E2E model is correct, and the misbehavior comes from improper tuning of DNN parameters. Therefore, we focus on repairing weights by following previous DNN repair works [6]–[10].

*Conclusion validity.* One threat of this type can come from the randomness of the search algorithm used in the repair. To mitigate this issue, we repeat each experiment 10 times, following the guideline of Arcuri and Briand [17], to ensure the reliability of the results. Moreover, still following [17], we have used suitable statistical tests (see §IV-G): Spearman correlation test to check correlation between open-loop and closed-loop metrics, Wilcoxon signed-rank test and Cohen's $d$ to compare the different components of the approach, and multi-factor ANOVA for sensitivity analysis.

*Internal validity.* An internal validity threat could come from the fact that the experimental results may be obtained by chance, for example due to implementation errors (i.e., *instrumentation threat*). To mitigate this issue, we have thoroughly checked the implementation of our approach. Another internal validity threat is that DNN repair may introduce regressions by turning originally positive inputs into negative ones. Since the repaired weights are shared across different inputs, improving the behavior on negative inputs may affect originally positive ones. Our goal is therefore not to eliminate this risk entirely, but to improve the behavior on negative inputs while limiting the degradation of positive inputs. To this end, our repair dataset includes both positive and negative inputs, and the fitness function is designed to consider both the repair of negative inputs and the preservation of positive inputs.

*External validity.* An external validity threat is the generalizability of E2ERep. To mitigate it, we select two different types of E2E models, UniAD and VAD, which are both applied in the E2E frameworks and have different model architectures.

## VIII. Related Work

DNN repair has emerged as a promising approach to fix known failures and ensure the quality of deep learning models. Over the years, it has been extensively studied and various methodologies have been established. Among these approaches, those using evolutionary algorithms are widely adopted due to their interpretability and scalability. As described in §I, these approaches operate in two phases: *fault localization* and *search-based repair*. Regarding fault localization, different approaches have been proposed. Eniser et al. [21] propose DeepFault, which adapts ideas of Spectrum-Based Fault Localization (a technique of classic software debugging) to neural networks. DeepFault leverages neuron activation to measure the impact on the output, and then introduces new suspiciousness metrics to identify the faulty weights that are likely contributing to erroneous behavior.

These fault localization techniques provide an initial step for identifying candidate weights for repair; based on that, Sohn et al. [6] propose Arachne, a framework for search-based repair of DNN classifiers. Arachne prioritizes the weights relevant to incorrect outputs, but it introduces the metrics *Gradient Loss* and *Forward Impact* to quantify the impact of each weight on the final output of DNNs. Then, it selects the Pareto-optimal weights according to these two metrics, and tunes these weights by an optimization algorithm. The optimization problem consists of searching for alternative weight values that maintain the correct behaviors and fix the wrong ones.

Inspired by these approaches, subsequent research has extended the ideas for different purposes. Li Calsi et al. [7] propose DistrRep, which repairs safety-critical scenarios by generating category-specific patches and combining them based on expert-judged risk levels. Tokui et al. [10] propose NeuRecover, which selects candidate parameters through set-based analyses over training data, with their relevance assessed by comparing regressed and improved behaviors. Sun et al. [11] propose CARE, which employs causality analysis to identify "guilty" neurons and adjusts their associated weights using search-based optimization. Such neuron-level localization, however, could identify too many weights to repair, thereby enlarging the search space. Ma et al. [22] propose

VERE, a verification-guided repair technique that localizes and optimizes neurons through linear relaxation. However, VERE relies on expensive verification over formally specified input regions and repair properties, which may not well-scale with large E2E models. Chen et al. [23] propose IDNN, which mitigates model defects by isolating critical neurons rather than searching for alternative weight values to obtain an offline repaired model. Weng et al. [24] propose AtPatch, which dynamically redistributes anomalous attention maps at runtime without modifying model parameters. However, AtPatch targets attention-map intervention for defects manifested as anomalous attention, rather than weight-level fault localization and repair. In addition to DNN classifiers, Schneider et al. [13] also propose SEMSEGREP that aims to repair semantic segmentation networks; the proposed technique is also search-based, but it features a filtering method for selecting faulty weight selection that considers the metrics used in semantic segmentation contexts. We use their way of splitting the data into positive and negative as the baseline for our fault localization approach (see §III-A1a).

Compared with these approaches, our work considers a different application domain (i.e., E2E models), more complex DNN architectures (i.e., transformer-based), and a different task (i.e., the planning of the future trajectory). These differences lead to the design of a new fault localization approach, and different fitness functions for the search-based repair.

Recently there is also a growing interest in repairing DNNs in the context of cyber-physical systems (CPS). Lyu et al. [25] propose CONTRREP, which extends search-based repair for neural network controllers and designs proper fitness functions by leveraging system specifications. Lyu et al. [26] also propose a different fault localization approach, which provides different strategies for different granularities of the targeted DNN components. Compared to the E2E models we consider in our work, these works target small DNN controllers that take low-dimensional inputs and give simple control commands such as acceleration or deceleration; therefore, those approaches cannot be directly applied in our problem settings.

## IX. CONCLUSION

E2E models are a new type of models for autonomous driving that promise to solve some issues of the modular architecture, and that have drawn attention from automotive companies. Despite their advantages, guaranteeing their correctness is still challenging. In this paper, we propose E2EREP, a repair approach tailored for E2E models. E2EREP introduces a new fault localization technique specific to the regression task carried out by these models, and a search-based repair guided by three newly introduced fitness functions.

The experimental results over two representative transformer-based E2E models have provided evidence that improving the performance of E2E models w.r.t. open-loop metrics is possible. However, experiments have also shown that open-loop performance does not always lead to better closed-loop performance at the system level.

Still, discussion with our industry partner revealed that using closed-loop metrics during repair is not possible, as this would require a time cost that does not fit with the industrial needs of fast development and prototyping. Therefore, we identified some future research directions that should be investigated to obtain a repair approach for E2E models that is both effective in terms of closed-loop performance and efficient.

## REFERENCES


[1] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020.

[2] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li, "End-to-end autonomous driving: Challenges and frontiers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 12, pp. 10 164–10 183, Dec. 2024.

[3] Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, W. Wang, L. Lu, X. Jia, Q. Liu, J. Dai, Y. Qiao, and H. Li, "Planning-oriented autonomous driving," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 17 853–17 862.

[4] B. Jiang, S. Chen, Q. Xu, B. Liao, J. Chen, H. Zhou, Q. Zhang, W. Liu, C. Huang, and X. Wang, "VAD: Vectorized scene representation for efficient autonomous driving," in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 8306–8316.

[5] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "Hidden technical debt in machine learning systems," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 2503–2511.

[6] J. Sohn, S. Kang, and S. Yoo, "Arachne: Search-based repair of deep neural networks," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 4, May 2023.

[7] D. Li Calsi, M. Duran, X. Zhang, P. Arcaini, and F. Ishikawa, "Distributed repair of deep neural networks," in *2023 IEEE Conference on Software Testing, Verification and Validation (ICST)*, 2023, pp. 83–94.

[8] H. Zhang and W. K. Chan, "Apricot: A weight-adaptation approach to fixing deep learning models," in *Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '19. IEEE Press, 2019, pp. 376–387.

[9] X. Ren, J. Chen, F. Juefei-Xu, W. Xue, Q. Guo, L. Ma, J. Zhao, and S. Chen, "DARTSRepair: Core-failure-set guided DARTS for network robustness to common corruptions," *Pattern Recognition*, vol. 131, p. 108864, 2022.

[10] S. Tokui, S. Tokumoto, A. Yoshii, F. Ishikawa, T. Nakagawa, K. Munakata, and S. Kikuchi, "NeuRecover: Regression-controlled repair of deep neural networks with training history," in *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2022, pp. 1111–1121.

[11] B. Sun, J. Sun, L. H. Pham, and J. Shi, "Causality-based neural network repair," in *Proceedings of the 44th International Conference on Software Engineering*, ser. ICSE '22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 338–349.

[12] X. Jia, Z. Yang, Q. Li, Z. Zhang, and J. Yan, "Bench2Drive: towards multi-ability benchmarking of closed-loop end-to-end autonomous driving," in *Proceedings of the 38th International Conference on Neural Information Processing Systems*, ser. NIPS '24. Red Hook, NY, USA: Curran Associates Inc., 2024.

[13] S. Schneider, T. Sujovolsky, P. Arcaini, F. Ishikawa, and T. V. Truong Duy, "Filter-based repair of semantic segmentation in safety-critical systems," in *2025 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2025, pp. 349–359.

[14] K.-H. Chang, "Chapter 5 - Multiobjective optimization and advanced topics," in *Design Theory and Methods Using CAD/CAE*. Boston: Academic Press, 2015, pp. 325–406.

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

[16] "DriveTransformer: Unified transformer for scalable end-to-end autonomous driving," in *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[17] A. Arcuri and L. Briand, "A practical guide for using statistical tests to assess randomized algorithms in software engineering," in *Proceedings of the 33rd International Conference on Software Engineering*, ser. ICSE '11. New York, NY, USA: Association for Computing Machinery, 2011, pp. 1–10.

[18] P. Schober, C. Boer, and L. A. Schwarte, "Correlation coefficients: appropriate use and interpretation," *Anesthesia & analgesia*, vol. 126, no. 5, pp. 1763–1768, 2018.

[19] B. Kitchenham, L. Madeyski, D. Budgen, J. Keung, P. Brereton, S. Charters, S. Gibbs, and A. Pohthong, "Robust statistical methods for empirical software engineering," *Empirical Software Engineering*, vol. 22, pp. 579–630, 2017.

[20] J. T. Richardson, "Eta squared and partial eta squared as measures of effect size in educational research," *Educational research review*, vol. 6, no. 2, pp. 135–147, 2011.

[21] H. F. Eniser, S. Gerasimou, and A. Sen, "DeepFault: Fault localization for deep neural networks," in *Fundamental Approaches to Software Engineering*. Cham: Springer International Publishing, 2019, pp. 171–191.

[22] J. Ma, P. Yang, J. Wang, Y. Sun, C.-C. Huang, and Z. Wang, "VeRe: Verification guided synthesis for repairing deep neural networks," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24. New York, NY, USA: Association for Computing Machinery, 2024.

[23] J. Chen, J. Wang, Y. Sun, P. Cheng, and J. Chen, "Isolation-based debugging for neural networks," in *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2024. New York, NY, USA: Association for Computing Machinery, 2024, pp. 338–349.

[24] S. Weng, Y. Feng, J. Li, Y. Yin, X. Xie, and J. Liu, "AtPatch: Debugging transformers via hot-fixing over-attention," *CoRR*, vol. abs/2601.21695, 2026.

[25] D. Lyu, Z. Zhang, P. Arcaini, F. Ishikawa, T. Laurent, and J. Zhao, "Search-based repair of DNN controllers of AI-Enabled Cyber-Physical Systems guided by system-level specifications," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '24. New York, NY, USA: Association for Computing Machinery, 2024, pp. 1435–1444.

[26] D. Lyu, Z. Zhang, P. Arcaini, X. Zhang, F. Ishikawa, and J. Zhao, "SpectAcle: Fault localisation of AI-Enabled CPS by exploiting sequences of DNN controller inferences," *ACM Trans. Softw. Eng. Methodol.*, Nov. 2024.