Supporting DNN Safety Analysis and Retraining through Heatmap-based Unsupervised Learning

Hazem Fahmy, Fabrizio Pastore, *Member, IEEE,* Mojtaba Bagherzadeh *Member, IEEE,*
and Lionel Briand, *Fellow, IEEE*

***Abstract*—Deep neural networks (DNNs) are increasingly important in safety-critical systems, for example in their perception layer to analyze images. Unfortunately, there is a lack of methods to ensure the functional safety of DNN-based components.**

We observe three major challenges with existing practices regarding DNNs in safety-critical systems: (1) scenarios that are underrepresented in the test set may lead to serious safety violation risks, but may, however, remain unnoticed; (2) characterizing such high-risk scenarios is critical for safety analysis; (3) retraining DNNs to address these risks is poorly supported when causes of violations are difficult to determine.

To address these problems in the context of DNNs analyzing images, we propose HUDD, an approach that automatically supports the identification of root causes for DNN errors. HUDD identifies root causes by applying a clustering algorithm to heatmaps capturing the relevance of every DNN neuron on the DNN outcome. Also, HUDD retrains DNNs with images that are automatically selected based on their relatedness to the identified image clusters.

We evaluated HUDD with DNNs from the automotive domain. HUDD was able to identify all the distinct root causes of DNN errors, thus supporting safety analysis. Also, our retraining approach has shown to be more effective at improving DNN accuracy than existing approaches.

***Index Terms*—DNN Explanation, DNN Functional Safety Analysis, DNN Debugging, Heatmaps**

I. Introduction

DEEP Neural Networks (DNNs) are common building blocks in many modern software systems. This is true for cyber-physical systems, where DNNs are commonly used in their perception layer, and common in the automotive sector, where DNN-based products have shown to effectively automate difficult tasks. For example, DNNs are used in Advanced Driver Assistance Systems (ADAS) to automate driving tasks such as emergency braking or lane changing [1], [2]. The rise of DNN-based systems concerns manufacturers that produce intelligent car components [3], [4]. This is the case of IEE [3], our industry partner in this research, who develops in-vehicle monitoring systems such as drowsiness detection and gaze detection systems [5].

DNNs consist of layers of hundreds of neurons transforming high-dimensional vectors through linear and non-linear activation functions, whose parameters are learned during training. Such structure prevents engineers from understanding the rationale of predictions through manual inspection of DNNs and, consequently, inhibits software quality assurance practices that rely on the analysis and understanding of the system logic. Such practices include failure root cause analysis and program debugging, which are the target of this paper.

A root cause is *a source of a defect such that if it is removed, the defect is decreased or removed* [6]. With DNN-based systems, root cause analysis consists in characterizing system inputs that lead to erroneous DNN results. For example, in image classification tasks, a root cause of DNN errors could be severe gender imbalance in the training set leading the DNN to label most female doctors as nurses; it might be detected after noticing that error-inducing inputs are characterized by doctors with long hair [7]. The DNN can be efficiently retrained after including in the training set additional images featuring these error-inducing characteristics.

When DNN-based systems are used in a safety-critical context, root cause analysis is required to support safety analysis. Indeed, safety standards, such as ISO26262 [8] and ISO/PAS 21448 [9], enforce the identification of the situations in which the system might be unsafe (i.e., provide erroneous and unsafe outputs) and the design of countermeasures to put in place (e.g., integrating different types of sensors). In the case of DNN-based systems, because of the complex structure of DNNs, the clear identification of unsafe situations is a challenge.

When inputs are images, which is our focus here, existing solutions for root cause analysis generate heatmaps that use colors to capture the importance of pixels in their contribution to a DNN result [7], [10]. By inspecting the heatmaps generated for a set of erroneous results, a human operator can determine that these heatmaps highlight the same objects, which may suggest the root cause of the problem (e.g., long hair [7]). Based on the identified root cause, engineers can then retrain the DNN using additional images with similar characteristics. Unfortunately, this process is expensive and error-prone because it relies on the visual inspection of many generated heatmaps. MODE goes beyond visual inspection and supports the automated debugging of DNNs through the identification of likely error-inducing images to be used for retraining [11]. However, MODE cannot support safety analysis since it does not provide support to identify plausible and distinct root causes leading to DNN errors.

Manuscript received XX YY, 20ZZ; revised MONTH ZZ, 20ZZ.

H. Fahmy and F. Pastore are affiliated with the SnT Centre for Security, Reliability and Trust, University of Luxembourg, JFK 29, L1855, Luxembourg. (e-mail: hazem.fahmy@uni.lu; fabrizio.pastore@uni.lu).

M. Bagherzadeh is affiliated with the school of EECS, University of Ottawa, Ottawa, ON K1N 6N5, Canada (e-mail: m.bagherzadeh@uottawa.ca).

L. Briand holds shared appointments with the SnT Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg and the school of EECS, University of Ottawa, Ottawa, ON K1N 6N5, Canada (e-mail: lbriand@uottawa.ca).

To alleviate the limitations above, we propose Heatmap-based Unsupervised Debugging of DNNs (HUDD). HUDD relies on hierarchical agglomerative clustering [12] combined with a specific heatmap-based distance function to identify clusters of error-inducing images with similar heatmaps for internal layers. Since heatmaps capture the importance of neurons regarding their contribution to the DNN result, error-inducing images with similar heatmaps should share characteristics that drive the generation of erroneous DNN results. Each cluster should thus characterize a distinct root cause for the observed DNN errors, even in cases where such causes are infrequent. Images in such clusters should then help identify clear and distinct root causes and can serve as a basis for efficient and effective retraining. We focus on internal DNN layers because they act as an abstraction over the inputs (e.g., ignore image background).

More precisely, HUDD relies on the computed clusters to identify new images to be used to retrain the DNN. Given a potentially large set of collected or generated unlabeled images, HUDD selects the subset of images that are closer to the identified clusters according to a heatmap-based distance. These images are then labeled by engineers and used to retrain the network. Labeling only a subset of images reduces retraining cost.

We performed an empirical evaluation on six DNNs. Our empirical results show that HUDD can automatically and accurately identify the different root causes of DNN errors. Also, our results suggest that the HUDD retraining process, improves DNN accuracy up to 30.24 percentage points and is more effective than baseline approaches.

The paper is structured as follows. Section II provides the context and motivation for this work. Section III summarizes background information. Section IV presents the proposed approach in details. Section V reports on the results of our empirical evaluation. Section VI discusses related work. Section VII concludes the paper.

II. Motivation and Context

In this section, we introduce the practical context of our research, which in short is the safety analysis and debugging of DNN-based automotive systems. We explain why automated root cause analysis is necessary to enable functional safety analysis. Also, we show how DNN accuracy improvement can be facilitated by the automated characterization and identification of *error-inducing inputs* (i.e., inputs that make the DNN generate erroneous results). Though the issues raised below and many of our insights are not specific to automotive systems, but also relevant to many cyber-physical systems in general, this is the practical domain and context in which this work took place.

A. DNN-based automotive systems

Our work is motivated by the challenges encountered in industry sectors developing safety-critical, cyber-physical systems, such as the automotive sector. For example, this is the case for IEE [3], a supplier of sensing solutions active in the automotive market and the provider of our case studies. In particular, IEE develops a gaze detection system (GDS) which uses DNNs to determine the gaze direction of the driver, from images captured by a camera on the instrument panel of the car.

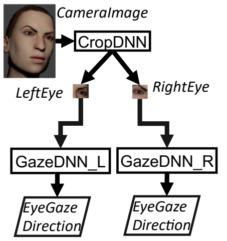


Fig. 1: DNN-based system for gaze detection.

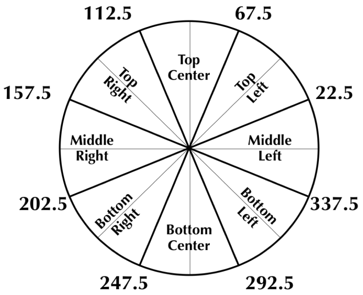


Fig. 2: Gaze directions.

IEE has been evaluating the feasibility of different GDS system architectures. Figure 1 shows an architecture consisting of three DNNs (i.e., CropDNN, GazeDNN_L, GazeDNN_R). CropDNN identifies face landmarks that enable the cropping of images containing the eyes only. GazeDNN_L and GazeDNN_R classify the gaze direction into eight classes (i.e., TopLeft, TopCenter, TopRight, MiddleLeft, MiddleRight, BottomLeft, BottomCenter, and BottomRight).

To reduce training costs, IEE relies on training sets containing images that are collected from driving scenes and images generated by simulation software. Simulators are used to reduce the costs related to data collection and data labeling. Indeed, models of the dynamics of real-world elements (e.g., eyeballs) are used to generate, in a controlled way through the selection of parameter values, hundreds of images in a few hours [13]. Further, and this is important in terms of cost saving, simulation enables the automated labeling of images by analyzing model parameters. However, while simulator images alleviate the costs of training, testing ultimately requires real-world images as well since simulators do not exhibit perfect fidelity.

In our experiments with IEE, we rely on the UnityEyes simulator to generate eye images [13]. UnityEyes combines a generative 3D model of the human eye region with a real-time rendering framework based on Blender [14]. We determine the gaze direction label from the gaze angle parameter provided by UnityEyes, based on predefined gaze ranges depicted in Figure 2. For example, we assign the label *TopCenter* when the gaze angle is between 67.5 and 112.5 degrees.

Additional DNN-based systems under development at IEE, which we used as cases studies, are presented in Section V.

B. Debugging of DNN-based Systems

IEE engineers train the DNNs that compose their systems by following the standard machine learning process depicted in Figure 3-a. They first train the DNN using a training set with labeled images (Step A) and then execute the DNN against a labeled test set (Step B). This process enables engineers to evaluate the DNN accuracy (e.g., the percentage of images leading to correct results).

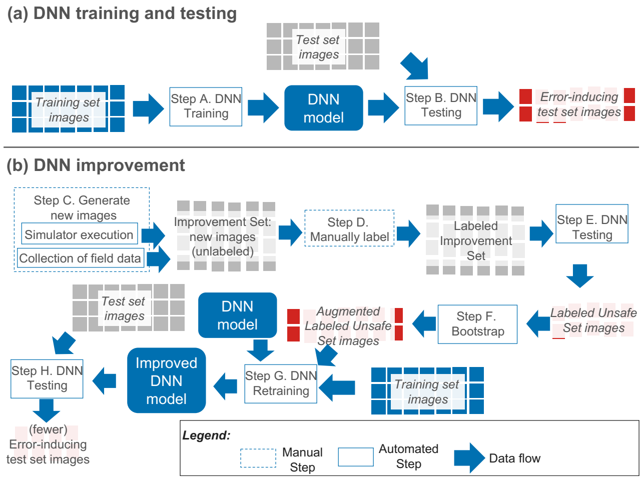


Fig. 3: Training and debugging DNNs.

When the accuracy of the system is not adequate, engineers typically improve the DNN by augmenting the training set with error-inducing images. This process is depicted in Figure 3-b. First, engineers generate a set of new images to be used to retrain the DNN (Step C). We call this set of images *improvement set*. The improvement set generally consists of images collected from the field since these tend to be error-inducing when DNNs have been trained using simulator images. Real-world images must be manually labelled (Step D). The DNN model is tested with the improvement set and images that lead to DNN errors are identified (Step E). This set of error-inducing (unsafe) images is considered to retrain the DNN (Step G), using as initial configuration for DNN weights the ones in the previously trained model.

To improve the DNN, it is necessary to process a sufficiently large number of unsafe images. For this reason, the number of unsafe images can be augmented by applying *bootstrap resampling* (i.e., by replicating samples in the unsafe set [15]) till a target is achieved (Step F). Finally, the improved DNN can be assessed on the test set (Step H).

In general, generating a sufficiently diverse set of error-inducing inputs that include all possible causes of DNN errors is very difficult. Further, when the labeling of such images is manual, the costs of labeling becomes prohibitive and DNN improvement is hampered. For this reason, **automatically characterizing images that are likely to lead to DNN errors** would allow the image generation or selection process to target specific types of images and increase the efficiency of the DNN retraining process.

C. Functional safety analysis

Like many other organizations in the automotive domain, IEE products must comply with the functional safety standards ISO 26262 and ISO/PAS 21448. Functional safety is addressed by identifying, for each component of the product (e.g., the DNNs of the GDS), the *unsafe conditions* that could lead to hazards, by identifying countermeasures (e.g., redundant components), and by demonstrating that these unsafe conditions are unlikely to occur.

ISO/PAS 21448, specifically targeting autonomous systems, recommends to determine unsafe conditions by following the traditional DNN testing process depicted in Figure 3-a and by manually inspecting the error-inducing images to look for root causes of DNN errors. In a DNN context, *unsafe conditions thus correspond to root causes of DNN errors*.

According to ISO/PAS 21448, engineers can set a quantitative target for accuracy evaluation to demonstrate that unsafe situations are unlikely. However, ISO/PAS 21448 also points out that quantitative targets are not sufficient and that engineers remain liable for potentially hazardous scenarios missing from the test set.

In addition, the manual identification of unsafe conditions is error-prone. For example, engineers may overlook unsafe conditions that are underrepresented in the test set. Also, such conditions may lead to a biased estimate of the accuracy of the DNN. For example, UnityEyes generates eye images where the horizontal angle of the head is determined based on a uniform distribution, between 160 (head turned right) and 220 degrees (head turned left). As a result, very few images with an angle of 160 or 220 degrees are generated and, though it may be an unsafe condition (i.e., one eye is barely visible and the gaze direction prediction may be inaccurate), experiments based on test sets generated with UnityEyes may suggest that the DNN is on average very accurate. It is, however, important for engineers to know that such a DNN, in some rarely occurring cases in the test set, is unsafe when the driver turns his head while driving.

In summary, accuracy estimation results depend on the test set, which may not include all unsafe conditions in a representative or balanced manner. Automated root cause analysis helps making sure, through clustering, that even rare, unsafe conditions are made visible to the analyst, especially when safety analysis time is limited. In other words, clustering based on heatmaps makes safety analysis robust, to some extent, to imperfect test sets.

III. Background

A. DNN Explanation and Heatmaps

Approaches that aim to explain DNN results have been developed in recent years [16]. Most of these concern the generation of heatmaps that capture the importance of pixels in image predictions. They include black-box [17], [18] and white-box approaches [10], [7], [19], [20], [21]. Black-box approaches generate heatmaps for the input layer and do not provide insights regarding internal DNN layers. In this paper, we therefore resort to white box approaches which rely on the backpropagation of the relevance score computed by the DNN [10], [7], [19], [20], [21]; Castanon et al. provide an overview of the state of the art [22]. In this paper, we rely on Layer-Wise Relevance Propagation (LRP) [10] because of the limitations of other approaches. First, solutions [21] backpropagating only the difference in activations between the different classes may compromise clustering since they do not account for information about all available neurons but only the ones related to the predicted output class. Deconvolutional networks [19] and guided backpropagation [20] lead to sparse heatmaps that do not fully explain the DNN result [23]. Grad-CAM [7] does not work with convolutional DNN layers. In contrast, LRP generates precise, non-sparse heatmaps for all the DNN layers because it takes into account all the different factors affecting the relevance of a neuron, which include the DNN structure and the neuron activations.

LRP redistributes the relevance scores of neurons in a higher layer to those of the lower layer. Assuming j and k to be two consecutive layers of the DNN, LRP propagates the relevance scores computed for a given layer k into a neuron of the lower layer j. It has been theoretically justified as a form of Taylor decomposition [24].

Figure 4 illustrates the execution of LRP on a fully connected network used to classify inputs. LRP analyzes the data processed by a DNN and can be applied to any DNN architecture. In the forward pass, the DNN receives an input and generates an output (e.g., classifies the gaze direction as TopLeft) while keeping trace of the activations of each neuron. The heatmap is generated in a backward pass.

In Figure 4, blue lines show that the DNN score of the selected class is backpropagated to lower layers. Plain lines show the connections concerned by the propagation formula used to compute the relevance (R_{ji}) of neuron i at layer j from all the connected neurons in layer k. $R_{ji} = \sum_l(\frac{z_{ji\_kl}}{\sum_i z_{ji\_kl}} * R_{kl})$, where z_{ji_kl} captures the extent to which neuron ji has contributed to make neuron kl relevant, and R_{kl} captures the relevance of neuron l at layer k. For example, for linear layers, $z_{ji\_kl} = a_{ji} * w^+_{ji\_kl}$, where a_{ji} is the activation of neuron i at layer j and $w^+_{ji\_kl}$ is the value of the weight on the connection between neuron ji and neuron ki, considering positive weights only. The denominator is used to redistribute the relevance received by a neuron to the lower layer proportionally to relative contributions. In our experiments, we have applied LRP to Convolutional Neural Networks (CNNs [25]), for classification tasks, and Hourglass Neural Networks [26], for regression tasks. We rely on the LRP and z_{ji_kl} implementation provided by LRP authors [27].

The heatmap in Figure 4 shows that the result computed by the DNN was mostly influenced by the pupil and part of the eyelid, which are the non-white parts in the heatmap.

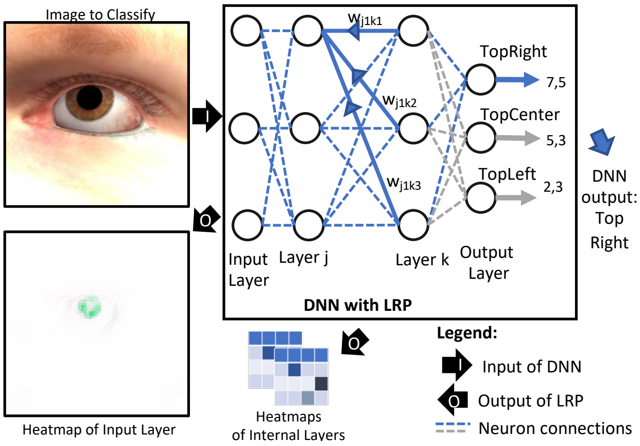


Fig. 4: Layer-Wise Relevance Propagation.

An additional key benefit of LRP is that it enables the computation of *internal heatmaps*, i.e., heatmaps for the internal layers of the DNN, based on the relevance score computed for every neuron in every layer. An internal heatmap for a layer k consists of a matrix with the relevance scores computed for all the neurons of layer k.

B. Unsupervised Learning

Unsupervised learning concerns the automated identification of patterns in data sets without pre-existing labels. In this paper, we rely on hierarchical agglomerative clustering (HAC) [12] to identify groups of error-inducing images with similar characteristics.

HAC is a bottom up approach in which each observation starts in its own cluster and pairs of clusters are iteratively merged into a sequence of nested partitions. The input of HAC is a matrix capturing the distance between every observation pair. The grouping that occurs at each step aims to minimize an objective function. In HAC, widely adopted objective functions are (1) the error sum of squares within clusters (i.e.,Ward's linkage method [28]), to help minimize within-cluster variance, (2) the average of distances between all pairs of elements belonging to distinct clusters (i.e., average linkage [29]), to help maximize diversity among clusters, and (3) the shortest distance between a pair of elements in two clusters (i.e., single linkage [30]), to merge clusters that are closer for at least one element.

HAC leads to a hierarchy of clusters that can be represented as a dendrogram. To identify the *optimal number of clusters*, we rely on the knee-point method [31], a recent approach that has been applied in different contexts, including fault localization [32] and performance optimization [33]. It automates the *elbow method* heuristics [34], which is commonly used in cluster analysis and consists of plotting the variance within a cluster as a function of the number of clusters, and picking the curve's elbow as the number of clusters to use. The knee-point method automates it by fitting a spline to raw data through univariate interpolation and normalizing min/max values of the fitted data. The knee-points are the points at which the curve differs most from the straight line segment connecting the first and last data point.

We chose HAC over K-means [35] since the latter requires the number of clusters to be known or predicted. In our case, K-means would thus need to be repeatedly executed in order to determine the optimal number of clusters; in the case of HAC, instead, the generated dendrogram provides all the required information in a single run. Further, HAC does not require the computation of cluster centroids [36], which is particularly expensive when differences between observations are computed from large matrices [37]. To more formally motivate our choice, we compare the worst case time complexity of HAC and K-means. HAC's running time is in $\mathcal{O}\left(\frac{d\cdot n\cdot(n-1)}{2} + n^2\right) \approx \mathcal{O}\left(d \cdot n^2\right)$, with n being the number of instances to cluster, and d being number of features to consider during clustering (i.e., the entries of the heatmap matrix, see Section IV-A). In other words, this complexity depends on the cost of computing the distance matrix, which is equal to the number of features multiplied by the number of image pairs (first addend), and the time complexity of HAC with Ward linkage (second addend [36]). The worst case time complexity of a single iteration for K-means is $\mathcal{O}\left(n^{k \cdot d}\right)$, with k being the number of clusters to consider [38], [39]. In our experiments, n lies in the range [506-5371], the *optimal number of clusters* is between 11 and 20 (see Section V-B1), and d is very large for DNN convolutional layers (e.g., 169×256, see Section IV-A). These numbers show that the worst case complexity of K-means is much larger than that of HAC, which further motivates our choice. We leave to future work the empirical evaluation of K-means and other clustering solutions.

IV. The HUDD Approach

Figure 5 provides an overview of our approach, HUDD, which provides two main contributions: (1) it automatically identifies the root causes of DNN errors, (2) it automatically identifies unsafe images for retraining the DNN. HUDD consists of six steps, described below.

In Step 1, HUDD performs heatmap-based clustering. This is a core contribution of this paper and consists of three activities: (1) generate heatmaps for the error-inducing test set images, (2) compute distances between every pair of images using a distance function based on their heatmaps, and (3) execute hierarchical agglomerative clustering to group images based on the computed distances. Step 1 leads to the identification of root cause clusters, i.e., clusters of images with a common root cause for the observed DNN errors.

In Step 2, engineers inspect the root cause clusters (typically a small number of representative images) to identify unsafe conditions, as required by functional safety analysis. The inspection of root cause clusters is an activity performed to gain a better understanding of the limitations of the DNN and thus introduce countermeasures for safety purposes (see Section II-C), if needed. However, the inspection of root cause clusters has no bearing on the later steps of our approach, including retraining.

In Step 3, engineers rely on real-world data or simulation software to generate a new set of images to retrain the DNN, referred to as the *improvement set*.

In Step 4, HUDD *automatically* identifies the subset of images belonging to the improvement set that are likely to lead to DNN errors, referred to as the *unsafe set*. It is obtained by assigning the images of the improvement set to the root cause clusters according to their heatmap-based distance.

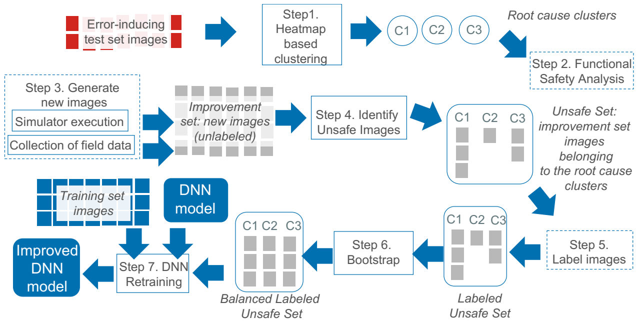

Error-inducing test set images
Step1. Heatmap based clustering
C1
C2
C3
Root cause clusters
Step 2. Functional Safety Analysis
Step 3. Generate new images
Simulator execution
Collection of field data
Improvement set: new images (unlabeled)
Step 4. Identify Unsafe Images
C1
C2
C3
Unsafe Set: improvement set images belonging to the root cause clusters
Training set images
DNN model
Improved DNN model
Step 7. DNN Retraining
C1 C2 C3
Balanced Labeled Unsafe Set
Step 6. Bootstrap
C1 C2 C3
Labeled Unsafe Set
Step 5. Label images

Fig. 5: Overview of HUDD (legend in Fig. 3).

In Step 5, engineers manually label the images belonging to the unsafe set, if needed (e.g., in the case of real images). Different from traditional practice (see Figure 3-b), HUDD requires that engineers label only a small subset of the improvement set.

In Step 6, to improve the accuracy of the DNN for every root cause observed, regardless of their frequency of occurrence in the training set, HUDD balances the labeled unsafe set using a bootstrap resampling approach.

In Step 7, the DNN model is retrained by relying on a training set that consists of the union of the original training set and the balanced labeled unsafe set.

Three out of the seven steps are manual (Steps 2, 3, and 5). However, these steps are also part of state-of-the-art solutions (see Section II-B). But in the case of HUDD the manual effort required in such steps is much more limited than in existing approaches. With HUDD, in Step 2, engineers inspect a few images per root cause clusters rather than the whole set of images, thus resulting in (a) significant cost savings (see Section V-B1) and (b) effective guidance towards the identification of root causes. In Step 5, with HUDD, engineers label only a subset of the improvement set that contains likely unsafe images identified by HUDD. Such unsafe images can be effectively used for retraining. Without HUDD, engineers would label a randomly selected subset of the improvements set, which would likely contain less unsafe images and thus be less effective during retraining (see Section 13). Finally, Step 3 is common practice and entails limited effort (e.g., buying field images or configuring a simulator).

The quality of HUDD results does not depend on the personal ability of engineers involved in manual steps; indeed, manual steps either concern the inspection of HUDD results or involve simple activities. The first contribution of HUDD (i.e., identify root causes of DNN errors) is provided by Step 1, which is fully automated. Step 2, which is manual, concerns the visual inspection of the generated clusters, does not require particular skills, and is part of state-of-the-art approaches. However, with HUDD, this step is facilitated by the quality of the generated clusters; for example, in Section V-B we demonstrate that HUDD generates root cause clusters presenting a common set of characteristics that are plausible causes of DNN errors, thus facilitating the identification of these root causes. The other manual steps (i.e., Step 3 and Step 5) are simple. In Step 3, engineers simply generate additional images (their selection is automated by HUDD in Step 4). In Step 5, engineers provide additional labels, which is an activity that, despite being time-consuming, can be assumed to be correct most of the time and is unavoidable when supervised learning (e.g., DNNs) is involved. The other steps leading to the second contribution of HUDD (i.e., identification of unsafe images and DNN retraining), are fully automated.

The following sections describe in detail all the steps of the approach, except Steps 3 and 5, which were introduced in Section II-B.

A. Heatmap-based clustering

HUDD is based on the intuition that, since heatmaps capture the relevance of each neuron on DNN results, error-inducing inputs sharing the same root cause should show similar heatmaps. For this reason, to identify the root causes of DNN errors, we rely on clustering based on heatmaps. Figure 6 provides an overview of our clustering approach.

For each error-inducing image in the test set, HUDD relies on LRP to generate heatmaps of internal DNN layers. Each heatmap captures the relevance score of each neuron in that layer.

A heatmap is a matrix with entries in $\mathbb{R}$, i.e., it is a triple (N, M, f) where $N, M \in \mathbb{N}$ and f is a map $[N] \times [M] \rightarrow \mathbb{R}$. We use the syntax $H[i, j]^L_x$ to refer to an entry in row i (i.e., $i < N$) and column j (i.e., $j < M$) of a heatmap H computed on layer L from an image x. The size of the heatmap matrix (i.e., the number of entries) is $N \cdot M$, with N and M depending on the dimensions of the DNN layer L. For convolution layers, N captures the number of neurons in the feature map, while M captures the number of feature maps. For example, the heatmap for the eighth layer of AlexNet has size 169×256 (convolution layer), while the the heatmap for the tenth layer has size 4096×1 (linear layer).

Since distinct DNN layers lead to entries defined on different value ranges [24], to enable the comparison of clustering results across different layers, we generate normalized heatmaps by relying on min-max normalization [40]. For a layer L, we identify min_L as the minimum value observed for all the heatmaps generated for a layer L, i.e.,

$$min_L \leq H[i, j]^L_x \; \forall \; i < N, j < M \tag{1}$$

with x being an image belonging to the unsafe test set. The maximum value max_L is derived accordingly. An entry $\tilde{H}[i, j]^L_x$ belonging to a normalized heatmap $\tilde{H}^T_x$ is derived as

$$\tilde{H}[i, j]^L_x = \frac{H[i, j]^L_x - min_L}{max_L - min_L} \tag{2}$$

The generated normalized heatmaps are used to build, for each DNN layer, a distance matrix that captures the distance between every pair of error-inducing image in the test set. The distance between a pair of images $\langle a, b \rangle$, at layer L, is computed as follows:

$$heatmapDistance_L(a, b) = EuclideanDistance(\tilde{H}^L_a, \tilde{H}^L_b) \tag{3}$$

where $\tilde{H}^L_x$ is the normalized heatmap computed for image x at layer L.

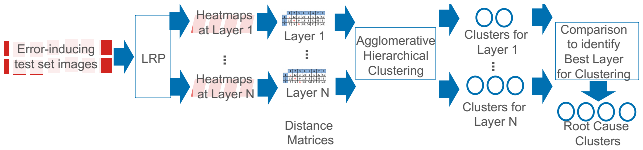


Fig. 6: Heatmap-based Clustering

$EuclideanDistance$ is a function that computes the euclidean distance between two $N \times M$ matrices according to the formula

$$EuclideanDistance(A, B) = \sqrt{\sum_{i=1}^{N} \sum_{j=1}^{M} (A_{i,j} - B_{i,j})^2} \tag{4}$$

where $A_{i,j}$ and $B_{i,j}$ are the values in the cell at row i and column j of the matrix.

Since we aim to generate clusters including images with similar characteristics, for each layer, we identify clusters of images by relying on the HAC algorithm with Ward linkage, which minimises within cluster variance (see Section III-B). We select the optimal number of clusters for a layer using the *knee-point method* applied to the weighted average intra-cluster distance.

In our context, clustering results are informative if they group together images that are misclassified for a same reason (i.e., if clusters are cohese) and if similar images belong to a same cluster (i.e., clusters are not fragmented). We determine cluster cohesion based on the weighted average intra-cluster distance ($WICD$), which we define according to the following formula:

$$WICD(L_l) = \frac{\sum_{j=1}^{|L_l|} \left(ICD(L_l, C_j) * \frac{|C_j|}{|C|} \right)}{|L_l|} \tag{5}$$

where L_l is a specific layer of the DNN, $|L_l|$ is the number of clusters in the layer L_l, ICD is the intra-cluster distance for cluster C_i belonging to layer L_l, $|C_j|$ is the number of elements in cluster C_j, while $|C|$ is the number of images in all the clusters.

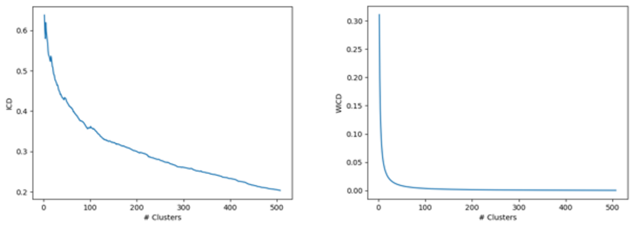
In Formula 5, $ICD(L_l, C_j)$ is computed as follows:

$$ICD(L_l, C_j) = \frac{\sum_{i=0}^{N_j} heatmapDistance_{L_l}(p^a_i, p^b_i)}{N_j} \tag{6}$$

where p_i is a unique pair of images in cluster C_j, and N_j is the total number of pairs it contains. The superscripts a and b refer to the two images of the pair to which the distance formula is applied.

In Formula 5, the factor $\frac{|C_j|}{|C|}$ normalizes the average ICD with respect to the relative size of the cluster. It helps determine the optimal number of clusters within a layer and enables the identification of the best clustering result across layers, as explained in the following paragraphs.

Since Ward linkage groups together elements that minimize within-cluster variance, an increase in the number of clusters (i.e., less elements per cluster) leads to a proportional decrease in the average ICD. In other words, the ICD slope is mild and smooth, which complicates the identification of the optimal number of clusters through the elbow method. By taking into account the relative size of the cluster, WICD helps determine when a larger number of clusters leads to suboptimal results, which happens when an increased number of cluster does not break down large clusters but rather divide small clusters into tiny ones (i.e., they are fragmented). Figure 7 shows the slope obtained for both ICD and WICD for a growing number of clusters; the plot for WICD clearly helps identify the sub-range on the X-axis leading to a drastic change in the slope, thus



(a) ICD

(b) WICD

Fig. 7: Slope of ICD and WICD for GazeDNN

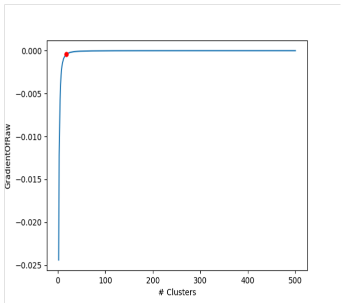


Fig. 8: Gradient and knee-point (in red) for Figure 7-b

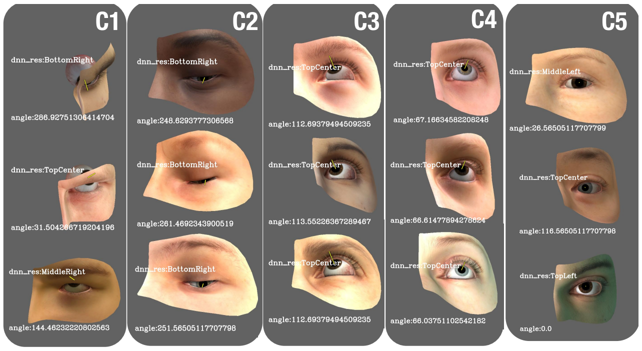


Fig. 9: Clustering results for GazeDNN.

enabling the identification of an optimal number of clusters beyond which WICD barely decreases.

To determine when WICD stops decreasing significantly, we rely on its derivative that we approximate by relying on the fourth order central difference method [41]. We then rely on the knee-point method to identify the point with the maximum curvature in the derivative [31]. Figure 8 shows an example knee-point automatically identified with our method.

HUDD selects the layer L_m with the minimal $WICD$. By definition, the clusters generated for layer L_m are the ones that maximize cohesion and we therefore expect them to group together images that present similar characteristics, suggesting root causes for DNN errors.

When comparing clusters for distinct layers, the normalization based on the relative size of the cluster (i.e., the factor $\frac{|C_j|}{|C|}$ in Equation 5) enables HUDD to penalize layers including large clusters with high ICD. These clusters group together images with heatmaps that are different from each other and thus may be associated with different root causes for DNN errors.

B. Root Causes Inspection

Root cause clusters are then inspected by engineers to determine unsafe conditions. For example, Figure 9 shows the clusters generated for the GazeDNN in Figure 1 on a test set with eye images generated by UnityEyes. To simplify the understanding of root causes, we printed the gaze angle on each image. Clusters C1 and C2 group together images that lead to DNN errors because the pupil is barely visible. In contrast, clusters C3 and C4 group images that are misclassified because the gaze angle is close to the classification threshold. Cluster C5, however, shows images that are misclassified because the training set labels are incomplete and do not capture the case of an eye looking middle center.

HUDD correctly handles both (1) the case in which erroneous DNN results with different output labels share the same root cause and (2) the case in which erroneous DNN results with the same output label are caused by distinct root causes. The first case is exemplified by cluster C5, which includes images that lead to different erroneous results (e.g., TopCenter or TopLeft) due to the same root cause (i.e., the eye is looking middle center but the DNN was not trained to detect it). The second case is exemplified by clusters C5 and C3, both including images erroneously classified as TopCenter. In cluster C3, this is due to the gaze angle close to the threshold with class TopRight whereas Cluster C5 includes images erroneously classified as TopCenter that are actually middle center.

In addition, the clusters in Figure 9 show that **HUDD identifies root causes that are associated with an incomplete training set** (e.g., borderline cases for gaze angle detected by C3 and C4) but also with **an incomplete definition of the predicted classes** (i.e., the middle center gaze detected by cluster C5 and the closed eyes detected by cluster C2) and **limitations in our capacity to control the simulator** (i.e., unlikely face positions detected by cluster C1). The first case is addressed by HUDD retraining procedures (i.e., Steps 4-7) whereas the other causes require that engineers modify the DNN (e.g., to add an output class) or improve the simulator.

To further simplify the inspection of root cause clusters, our toolset also generates a set of animated GIF images, one for each cluster [42]. Each generated GIF image shows all the images belonging to a cluster one after the other. Animated GIFs enable engineers to inspect a large number of images in a few seconds (e.g., we configure our tool to visualize 100 images in a minute) thus facilitating the detection of the common characteristics among them.

C. Identification of Unsafe Images

HUDD processes the improvement set to automatically identify potentially unsafe images. This is done by assigning improvement set images to root cause clusters while limiting the number of assigned images.

To assign images to clusters, HUDD relies on the *single linkage method* (see Section III-B). According to the single linkage method, an image y belonging to the improvement set IS, is assigned to the cluster containing the closest error-inducing image from the test set. We rely on single linkage since it has a desirable property: If applied to the error-inducing test set images used to generate the clusters, it ensures that every image is assigned to the cluster it belongs to. This is important since, in a realistic scenario where test set and improvement set images are collected from the field, it ensures the selection of unsafe images that are highly similar to error-inducing ones.

Unfortunately, single linkage alone is not sufficient to identify unsafe images. Since the root cause clusters capture only the unsafe portion of the input space, every image in the improvement set, including the safe ones, will be assigned to a root cause cluster. In other words, a safe image might be assigned to a cluster simply because it happens to be closer to an image belonging to this particular cluster.

To address this problem, we heuristically select, for every cluster, the images that are likely error-inducing by estimating the number of error-inducing images for each error-cause cluster in the improvement set. Since the improvement set is generally derived from the same population as the test set (e.g., real world images collected according to the same strategy as for the test set), we can assume that (1) the improvement set is characterized by the same accuracy as the test set and (2) the causes of DNN errors in the improvement set should follow the same distribution as the one observed in the test set (i.e., for every root cause cluster, we should observe a number of error-inducing images proportional to what observed in the test set).

We compute the number U_{C_i} of images to be selected for a root cause cluster C_i, as follows:

$$U_{C_i} = (|\mathit{TestSet}| * \mathit{sf}) * (1 - \mathit{TestSetAcc}) * \frac{|C_i|}{|C|} \tag{7}$$

The term $(|\mathit{TestSet}| * \mathit{sf}) * (1 - \mathit{TestSetAcc})$ estimates the number of error-inducing images that should be selected from the improvement set. The term $\frac{|C_i|}{|C|}$ indicates how to distribute these images across root cause clusters in order to preserve the proportion of the test set across clusters in the improvement set. The term $(|\mathit{TestSet}| * \mathit{sf})$ provides an upper bound for the unsafe set size as a proportion of the test set size, which is determined by the available budget for labelling. Indeed, $|\mathit{TestSet}|$ is the size of the test set, while sf is a selection factor in the range [0-1] (we use 0.3 in our experiments). The term $(1 - \mathit{TestSetAcc})$ indicates the proportion of error-inducing images that should be observed in the improvement set, based on assumption (1) above. Multiplied by the unsafe set upper bound, it ensures that we select a fraction of it. Finally, the term $\frac{|C_i|}{|C|}$ indicates the fraction of unsafe set images that should be assigned to the root cause cluster C_i. The term $|C_i|$ is the size of the cluster C_i. The term $|C|$ is the number of error-inducing images in the test set. Based on assumption (2) above, the fraction $\frac{|C_i|}{|C|}$ corresponds to the proportion of error-inducing images from the test set belonging to the root cause cluster C_i.

Figure 10 shows the pseudocode of our algorithm for identifying unsafe images. It requires the root cause clusters R, the identifiers of the images belonging to the improvement set (IS), the selection factor sf, the test set accuracy (acc_{TS}), the size of the test set ($size_{TS}$), and a distance matrix DM_{IS} with the distances between the images in IS and the images in the error-inducing test set, for the layer selected for the identification of root cause clusters. To speed up the identification of unsafe images, since we focus on one specific layer, we compute the distance matrix based on the heatmaps returned by LRP, not the normalized ones.

The algorithm works by assigning U_{C_i} improvement set images to each root cause clusters C_i. It ensures that every image is assigned to the closest cluster C_i that has not been already filled with U_{C_i} images. The algorithm also handles the presence of spurious clusters, that is, clusters that group together diverse images for which a common root cause cannot be clearly identified. Spurious clusters may be assigned with safe images or error-inducing images that should belong to other clusters. By relying on a ranking strategy for the assignment of images to clusters, we alleviate the effect of spurious clusters. When spurious clusters are fully assigned, the algorithm correctly assigns to their respective clusters all the remaining images, even if they are accidentally closer to the spurious cluster.

In Figure 10, Lines 2 to 3 compute, for every cluster, the value of U_{C_i} according to Equation 7. Lines 4 to 14, for every image, rank clusters, based on single linkage distance. This is performed by computing the distance of the image from each cluster (Lines 5 to 9) and then sorting clusters accordingly (Lines 9 to 14), such that the cluster in rank 1 is the closest one.

In Lines 15 to 21, the algorithm assigns every image to the closer cluster that is not already full. It iterates over the ranks (Line 15), and for each rank r, it loops over the improvement set images starting from the ones that are closer to a cluster (Line 16). If the cluster (i.e., $clusterId$, Line 17) is not already full (Line 19), it assigns the image to the cluster (Line 20). If the cluster is full (e.g., if it is a spurious cluster), the image is processed in the next iteration, i.e., when the algorithm tries to assign images to clusters ranked as $r + 1$. Note that, for each rank, every image is processed only once (Lines 18 and 21 delete processed images).

The algorithm returns an unsafe set with U_{C_i} images selected for each cluster C_i. The selected images are labeled by engineers when required (Step 6 in Figure 5) and then used for retraining.

D. DNN Retraining

HUDD retrains the DNNs by executing the DNN training process against a data set that is the union of the original training set and the labeled unsafe set. HUDD uses the available model to set the initial configuration for the DNN weights. The original training set is retained to avoid reducing the accuracy of the DNN for parts of the input space that are safe (i.e., showing no error in the test set).

HUDD balances the unsafe set with bootstrap resampling [15], i.e., it randomly duplicates the images belonging to the cluster until every cluster has the same size. This is done to

```
Require: (1) $R$, root cause clusters. (2) $IS$, set with the identifiers of the images
    belonging to the improvement set. (3) $sf$, the selection factor used in Equation 7.
    (4) $size_{TS}$ the size of the test set. (5) $acc_{TS}$ the accuracy for the test set. (6)
    $DM_{IS}$, distance matrix capturing the distance between images in the improvement
    set and images in TS.
Ensure: an associative array with the unsafe images associated to each root cause cluster

//Initialize the array that will contain all the images to be processed
$rankedClustersPerImage \leftarrow$ new associative array that will contain,
                        for every image, the IDs of clusters,
                        ranked based on their distance from the image
//Set the max number of images to be assigned to each cluster
for clusterID in R do
    $Uc[clusterID] \leftarrow (size_{TS} * sf) * (1 - acc_{TS}) * \frac{sizeOf(R[clusterID])}{sizeOf(R)}$
//For each image, rank clusters, based on HeatmapDistance
for $img$ in $IS$ do
    //Generate an associative array capturing, for every cluster,
    the distance of $img$ from the closest image of the cluster
    for $clusterID$ in R do
        $clusterDists \leftarrow$ new associative array to store the distance of $img$
                        from every cluster
        $closest \leftarrow$ use $DM_{IS}$ to identify the test set image that is closer to $img$
                        among the ones belonging to $clusterID$
        $clusterDists[clusterID] \leftarrow$ distance between $closest$ and $img$
    //Put clusters in the correct rank for $img$
    for rank in 1 .. $|R|$ do
        $clusterID \leftarrow$ position in $clusterDists$ containing the lowest value
        $rankedClustersPerImage[rank][img] \leftarrow$
                        $< clusterID, clusterDists[clusterID] >$
        set $clusterDists[clusterID]$ to undefined
//Assign images to clusters, trying to assign every image to the closer cluster first
for rank in 1 .. $|R|$ do
    $img \leftarrow$ the index $img$, in $rankedClustersPerImage[rank][img]$
                containing the lowest value, i.e., the image that is closer to any of
                the clusters
    //Save the ID of the cluster that is closer to $img$
    $clusterId \leftarrow rankedClustersPerImage[rank][img]$
    delete $rankedClustersPerImage[rank][img]$
    //Add the image to the cluster, if this is not already full
    if sizeOf ( $unsafeSet[clusterId]$ ) $<$ $Uc[clusterId]$ then
        add $img$ to $unsafeSet[clusterId]$
        //Remove the image from the array with the images to process
        delete $rankedClustersPerImage[rank][img]$ for all the ranks
Return $unsafeSet$
```

Fig. 10: Algorithm for the identification of unsafe images

maximize the chances of eliminating every root cause of error, even the ones that are rare (i.e., the ones for which we identify less unsafe set images). More formally, assuming $Max(|U_{C_i}|)$ being the size of the largest root cause cluster, bootstrap resampling ensures that every cluster contains $Max(|U_{C_i}|)$ members. The retraining process is expected to lead to an improved DNN model compared to that based on the original training set.

V. Empirical Evaluation

Our empirical evaluation aims to address the following research questions:

RQ1. Does HUDD enable engineers to identify the root causes of DNN errors? We aim to investigate whether images belonging to a same cluster, as generated by HUDD, present a common set of characteristics that are plausible causes of DNN errors.

RQ2. How does HUDD compare with traditional DNN accuracy improvement practices? We aim to investigate whether HUDD enables engineers to efficiently drive the retraining of a DNN compared with state-of-the-art approaches.

To perform our empirical evaluation, we have implemented HUDD as a toolset that relies on the PyTorch [43] and SciPy [44] libraries. Our toolset, case studies, and results are available for download [45].

A. Subjects of the study

To address RQ1, we need to objectively and systematically identify commonalities among images belonging to the same cluster. To do so, we rely on images generated using simulators as it allows us to associate each generated image to values of the configuration parameters of the simulator. These parameters capture information about the characteristics of the elements in the image and can thus be used to objectively identify the likely root causes of DNN errors.

We consider DNNs that implement the key features of gaze detection, drowsiness detection, headpose detection, and face landmarks detection systems under development at IEE. The gaze detection system (hereafter referred as GD) has been presented in Section II-A. The drowsiness detection system (OC) features the same architecture as the gaze detection system, except that the DNN predicts whether eyes are closed. The headpose detection system (HPD) receives as input the cropped image of the head of a person and determines its pose according to nine classes (straight, turned bottom-left, turned left, turned top-left, turned bottom-right, turned right, turned top-right, reclined, looking up). The face landmark detection system (FLD) receives as input the cropped image of the head of a person and determines the location of the pixels corresponding to 27 face landmarks delimiting seven face elements: nose ridge, left eye, right eye, left brow, right brow, nose, mouth. Each face element is delimited by several face landmarks.

GD, OC, and HPD follow the AlexNet architecture [46] which is commonly used for image classification. FLD, which addresses a regression problem, relies on an Hourglass-like architecture [26]. It includes 27 output neurons, each one predicting the position (i.e., pixel) of a distinct face landmark. Since a small degree of error in the detected landmarks is considered acceptable, the output of FLD is considered erroneous if the average distance of the identified landmarks from the ground truth is above four pixels. To apply HUDD to FLD, we generate heatmaps by backpropagating the relevance of the worst output neuron, i.e., the output neuron with the highest distance from the ground truth. Since face elements present very different characteristics, we apply the HUDD clustering algorithm seven times, once for each face element, by selecting images whose worst output neuron corresponds to the considered face element.

The first four rows of Table I provide details about the four DNNs described above. Column *Data Source* reports the name of the simulator generating the images used to train and test the network. GD and OC have been trained and tested with images generated by UnityEyes. Since classes need to be balanced in order to properly train the DNN, for OC, we selected a subset of images consisting of all the closed eyes and the same number of open eyes. For GazeDNN, this is not needed since UnityEyes selects the gaze angle according to a uniform distribution. HPD and FLD have been trained and tested with images generated using a simulator developed in-house by IEE. The IEE simulator relies on 3D face models built with the MakeHuman [47] plug-in for Blender [14]. To emulate car cameras, it generates grey images. HPD and FLD share the same training and test sets. To generate images, we used six face models for the training set, one for the test set. The use of different face models for training and testing emulates realistic scenarios in which the images processed in the field belong to persons different than the ones considered for training the DNN. Figure 11 shows examples of nine head poses generated with the same face model.

In Table I, column *Epochs* reports the number of epochs considered to train the network. All the DNNs have been trained for a number of epochs that was sufficient to achieve a training set accuracy above 80%. Columns *Training Set Size* and *Test Set Size* report the size of the training and test sets. Columns *Accuracy Training* and *Accuracy Test* indicate the accuracy obtained by the DNN when executed against images in the training and test sets. Though training set accuracy is above 87% for all the four DNNs, in the case of HPD and FLD, we observe a lower test set accuracy. This is due to the prediction task being more complex for HPD and FLD than for GD and OC; indeed, the DNNs for HPD and FLD are tested with images belonging to face models that are different than the ones used for training. This was not the case for GD and OC since UnityEyes does not provide the means to control face features and automatically selects them during simulation. In addition, in contrast to UnityEyes, the images generated with the IEE simulator using different face models are likely to be more diverse. Indeed, the six face models integrated in UnityEyes capture only the face area surrounding the eye (i.e., eyelid and a portion of the nose) while the face models of the IEE simulator capture the entire face. Consequently, when testing is based on new face models, it is more likely to lead to DNN errors in the case of HPD and FLD. The number of face models considered for training HPD and FLD is limited to six since the definition of a face model is an expensive manual task.

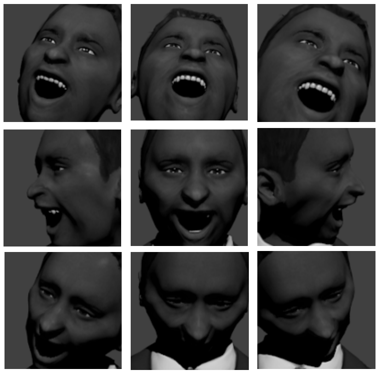


Fig. 11: Example of distinct head poses of the same person generated with the simulator based on MakeHuman/Blender.

TABLE I: Case Study Systems

DNN	Data Source	Training Set Size	Test Set Size	Epochs	Accuracy	
					Training	Test
GD	UnityEyes	61,063	132,630	10	96.84%	95.95%
OC	UnityEyes	1,704	4,232	10	87.38%	88.03%
HPD	Blender	16,013	2,825	10	94.45%	44.07%
FLD	Blender	16,013	2,825	10	88.97%	44.99%
OD	CelebA [48]	7916	5276	13	83.67%	84.11%
TSR	TrafficSigns [49]	29,416	12,631	12	92.64%	81.65%

Since HUDD can be applied to DNNs trained using simulator or real images, to address RQ2, which concerns the improvement achieved after retraining the DNN, we also considered additional DNNs trained using real-world images. We selected DNNs implementing traffic sign recognition (TSR), and object detection (OD), which are typical features of automotive, DNN-based systems. They are reported in the last two rows of Table I. TSR recognizes traffic signs in pictures. OD determines if a person wears eyeglasses. OD has been selected to compare results with MODE, a state-of-the-art retraining approach whose implementation is not available (see Section V-B4), but which is close in objective to HUDD. OD has been trained on the same dataset used for evaluating MODE but we selected a subset of the available images to balance classes (common practice). Though the original trained model is not available, we achieved the same accuracy as the one reported. The other two case studies considered in the MODE evaluation were discarded because they are either not representative (i.e., low accuracy) or lack information for enabling replication (i.e., description of inputs and outputs). TSR and OD follow the AlexNet architecture [46].

B. Measurements and Results

We refine RQ1 into three complementary subquestions (i.e., RQ1.1, RQ1.2, and RQ1.3), which are described in the following, along with the results obtained.

1) RQ1.1: Is the visual inspection of root cause clusters practically feasible?

Design and measurements. We discuss whether the number of clusters generated by HUDD is small enough to make visual inspection feasible.

Since this research question does not concern the quality of the generated clusters, we considered all the case studies, including the ones trained and tested with real-world images. For each case study system, we thus report the number of root cause clusters generated by HUDD. Also, under the assumption that engineers visually inspect five images for each root cause cluster, we discuss the ratio of error-inducing images that should be visually inspected when relying on HUDD. This ratio provides an indication of the time saved with respect to current practice (i.e., manual inspection of all the error-inducing images). A user study concerning the time savings introduced by HUDD is part of our future work.

Results.

Table II shows, for each case study, the total number of error-inducing images belonging to the test set, the number of root cause clusters generated by HUDD, and the ratio of error-inducing images that should be visually inspected when using HUDD.

For the respective DNNs, HUDD identifies 16 (GD), 14 (OC), 17 (HPD), 71(FLD), 14 (OD), and 20 (TSR) root cause clusters. For all the case studies except FLD, the number of root cause clusters generated is below or equal to 20. Assuming that engineers inspect few images (e.g., five) for

TABLE II: Root cause clusters generated by HUDD.

Case study	**# Error-inducing images**	**# Root cause clusters**	**Ratio of inspected images**
GD	5371	16	1.49%
OC	506	14	13.82%
HPD	1580	17	5.38%
FLD	1554	71	22.84%
OD	838	14	8.35%
TSR	2317	20	4.31%

TABLE III: Image parameters considered to address RQ1.1

DNN	**Parameter**	**Description**
GD/OC	Gaze Angle	Gaze angle in degrees.
	Openness	Distance between top and bottom eyelid in pixels.
	H_Headpose	Horizontal position of the head (degrees)
	V_Headpose	Vertical position of the head (degrees)
	Iris Size	Size of the iris.
	Pupil Size	Size of the pupil.
	PupilToBottom	Distance between the pupil bottom and the bottom eyelid margin.
	PupilToTop	Distance between the pupil top and the top eyelid margin.
	DistToCenter	Distance between the pupil center of the iris center. When the eye is looking middle center, this distance is below 11.5 pixels.
	Sky Exposure	Captures the degree of exposure of the panoramic photographs reflected in the eye cornea.
	Sky Rotation	Captures the degree of rotation of the panoramic photographs reflected in the eye cornea.
	Light	Captures the degree of intensity of the main source of illumination.
	Ambient	Captures the degree of intensity of the ambient illumination.
HPD	Camera Location	Location of the camera, in X-Y-Z coordinate system.
	Camera Direction	Direction of the camera (X-Y-Z coordinates).
	Lamp Color	RGB color of the light used to illuminate the scene.
	Lamp Direction	Direction of the illuminating light (X-Y-Z coordinates).
	Lamp Location	Location of the source of light (X-Y-Z coordinates).
	Headpose	Position of the head of the person (X-Y-Z coordinates). It is used to derive the ground truth.
FLD	X coordinate of landmark	Value of the horizontal axis coordinate for the pixel corresponding to the i^{th} landmark.
	Y coordinate of landmark	Value of the vertical axis coordinate for the pixel corresponding to the i^{th} landmark.

each cluster in order to determine plausible root causes, manual inspection based on HUDD appear to be practically feasible. In the case of FLD, the larger number of clusters is due to the identification of distinct root cause clusters for each face element; on average, we derive ten root cause clusters per element (within a [6 - 11] range). IEE engineers agreed that the manual inspection of a larger number of clusters is justified given the complexity of the case study.

In general, the ratio of error-inducing images that is inspected with HUDD is low, ranging from 1.49% (GD) to 22.84% (FLD), which shows that the analysis supported by HUDD saves a great deal of effort with respect to current practice (i.e., manual inspection of all the error-inducing images).

2) RQ1.2: Do the clusters generated by HUDD show a significant reduction in the variance of simulator parameters?

Design and measurements. This research question assesses if images belonging to the same cluster present similar characteristics. To address this research question, we rely on case studies trained and tested with simulator images. When images are generated with a simulator, images belonging to the same cluster should present similar values for a subset of the simulator parameters. In turn, this should result in a reduction of variance for these parameters in comparison to the entire error-inducing test set. For a cluster C_i, the rate of reduction in variance for a parameter p can be computed as follows:

$$RR^{p}_{C_i} = 1 - \frac{variance\ of p\ for\ the\ images\ in\ C_i}{variance\ of\ p\ for\ the\ entire\ error-inducing\ set}$$

Positive values for $RR^{p}_{C_i}$ indicate reduced variance.

Table III provides the list of parameters considered in our evaluation. In the case of GD and OC, we selected all the parameters provided by the simulator except the ones that capture coordinates of single points used to draw the pictures (e.g., eye landmarks) since these coordinates alone are not informative about the elements in the picture. However, we considered these coordinates to compute metrics that capture information about the scene in the image. We refer to such metrics as derived parameters. For example, we compute the distance between the bottom of the pupil and the bottom eyelid margin (*PupilToBottom* in Table III). It determines if the eye is in an unusual position, e.g., if the eye is at the bottom of the orbit. In the case of HPD, similarly to GD and OC, we considered the parameters provided by the simulator, excluding once again landmark coordinates. For parameters expressed with X-Y-Z coordinates, we considered the coordinate on each axis as a separate parameter. In the case of FLD, since a DNN error may depend on the specific shape and expression of the face being processed (i.e., on the specific position of a landmark), we considered the coordinates of the 27 landmarks on the horizontal and vertical axes as distinct parameters (54 parameters in total).

We compute the percentage of clusters showing reduction in variance for at least one of the parameters. Since we do not know a priori the number of parameters that capture common error causes, we consider variance reduction in one parameter to be sufficient. More precisely, we compute the percentage of clusters with a reduction in variance between 0.0 and 0.9, with incremental steps of 0.10. To answer positively our research question, a high percentage of the clusters should show a reduction in variance for at least one of the parameters.

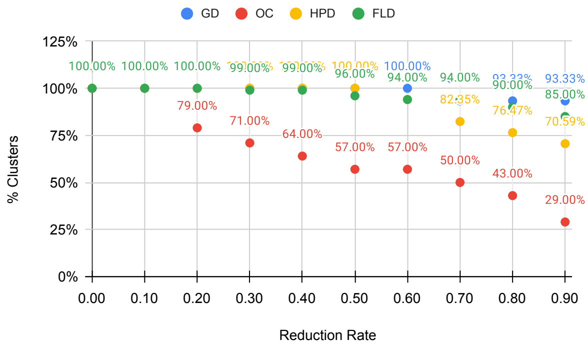


Fig. 12: RQ1.1: Clusters with at least one parameter showing a reduction rate above thresholds in the range (0.0 - 1.0).

TABLE IV: Safety parameters considered to address RQ1.3

DNN	Parameter	Unsafe values
GD,OC	Gaze Angle	Values used to label the gaze angle in eight classes (i.e., 22.5°, 67.5°, 112.5°, 157.5°, 202.5°, 247.5°, 292.5°, 337.5°).
	Openness	Value used to label the gaze openness in two classes (i.e., 20 pixels) or an eye abnormally open (i.e., 64 pixels).
	H_Headpose	Values indicating a head turned completely left or right (i.e., 160°, 220°)
	V_Headpose	Values indicating a head looking at the very top/bottom (i.e., 20°, 340°)
	DistToCenter	Value below which the eye is looking middle center (i.e., 11.5 pixels).
	PupilToBottom	Value below which the pupil is mostly under the eyelid (i.e., -16 pixels).
	PupilToTop	Value below which the pupil is mostly above the eyelid (i.e., -16 pixels).
HPD	Headpose-X	Boundary cases (i.e.,-28.88°,21.35°), values used to label the headpose in nine classes (-10°,10°), and middle position (i.e., 0°).
	Headpose-Y	Boundary cases (i.e.,-88.10°,74.17°), values used to label the headpose in nine classes (-10°,10°), and middle position (i.e., 0°).

Results. Figure 12 shows the percentage of clusters with variance reduction for at least one of the simulator parameters, at different reduction rates.

We can positively answer RQ1.1 since all the clusters present at least one parameter with a positive reduction rate (> 0 in Figure 12). Also, a very high percentage of the clusters (i.e., 57% for OC, 96% for FLD, and 100% for both GD and HPD) include at least one parameter with a reduction rate above or equal to 0.5, i.e., 50% reduction in variance. Expectedly, as the threshold considered for variance reduction increases, the percentage of clusters tends to decrease. However, a 0.9 threshold is still matched by 29% (OC) to 93.33% (GD) of the clusters (69.48% on average), a very high proportion, thus showing that most of the clusters should present a noticeable common characteristic.

3) *RQ1.3: Do parameters with high reduction in variance identify the plausible cause for DNN errors?*

Design and measurements. With RQ1.3, we ask whether the commonalities of the images belonging to the root cause clusters can help engineers determine the root causes of the DNN errors.

We expect DNN errors to be triggered in specific parts of the input space, each one capturing characteristics of the input images. To identify the input sub-spaces that are unsafe for our case studies, based on domain knowledge, we have identified a set of parameters (hereafter, *unsafe parameters*) for which it is possible to identify values (hereafter, *unsafe values*) around which, or below which, we are likely to observe a DNN error. However, for FLD, it was not possible to determine, a priori, a set of unsafe parameters that might affect the results and we had to leave out that case study. Indeed, the position of a landmark may depend on many factors including the shape of the face element (e.g., thick lips), the element position (e.g., mouth being open), the headpose, and the camera position. Since the IEE simulator does not export information about the shape and position of face elements, it was not possible to define a set of metrics capturing plausible error causes.

Table IV provides the list of unsafe parameters, along with the unsafe values identified. For example, for the Gaze Angle parameter, unsafe values consist of the boundary values used to label images with the gaze direction.

Root cause clusters that are explanatory should present at least one characteristic that is noticeable by the engineer, i.e., they should have at least one parameter with high (i.e., 50%) reduction in variance. In addition, at least one of the parameters with high variance reduction should be an unsafe parameter. Finally, the cluster average should be close to one unsafe value. For Gaze Angle, Openness, H_Headpose, V_Headpose, Headpose-X, and Headpose-Y, since unsafe values split the parameter domains into subranges, we determine that the cluster average is close to one unsafe value if the difference between them is below 25% of the subrange including the average value. For DistToCenter, PupilToBottom, and PupilToTop, we simply check if the average is below or equal to the unsafe value. Finally, we compute the percentage of clusters for which the conditions above hold. To answer positively to RQ1.3, this percentage should be high.

Results. In the case of GD, according to the conditions defined above, the percentage of clusters that identify the likely root cause of DNN errors is very high: 86.66% (13 out of 15). The identified unsafe parameters are *Angle*, *Openness*, and *DistToCenter*. For one cluster not meeting the conditions, the unsafe parameters (i.e., *DistToCenter*) have a reduction in variance of 44%, below the 50% threshold. This threshold is, however, arbitrary and a manual inspection of the cluster clearly shows that the commonality is the eye being abnormally open. The other non-compliant cluster shows pupils being partially masked by the eyelid; however, we could not define a measure to systematically capture this situation based on simulator parameters.

For OC, we obtain 57.14% (8 out of 14), with *Openness* and *X_Headpose* being the unsafe parameters. The remaining clusters are characterized either by thin almond eyes, an aspect of the simulation that is not controllable with parameters, and pupils being partially masked by the eyelid.

In the case of HPD, we obtain 88.24% (15 out of 17). For the two remaining clusters, the common characteristic is the presence of visible white teeth, which are not visible in training set images and may confuse the DNN since they stand out in grey-scale images. Based on the above observations, we respond positively to RQ1.3 since, in all cases, clusters are clearly associated with image characteristics that are plausible causes of errors.

4) *RQ2: How does HUDD compare to traditional DNN accuracy improvement practices?*

This research question aims to compare the accuracy improvements achieved by HUDD with the improvements achieved by baseline approaches, which do not rely on the automated selection of predicted unsafe images.

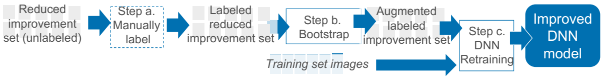

Reduced improvement set (unlabeled)
Step a. Manually label
Labeled reduced improvement set
Step b. Bootstrap
Augmented labeled improvement set
Step c. DNN Retraining
Improved DNN model
Training set images

Fig. 13: Baseline 2 (B2).

TABLE V: RQ2. Size of Images Set used for Retaining and Accuracy Improvement

	Size of Images Sets for Retraining								Accuracy	Accuracy (Accuracy improvement)			Delta	$\hat{A}_{12}$	
	HUDD			B1			B2		Original				wrt best	HUDD vs	
DNN	IS	US	BLUS	IS	LUS	ALUS	IS	ALIS	Model	HUDD	B1	B2	Baseline	B1	B2
GD	72500	1615	4192	1615	156.4	4192	1615	4192	95.95%	96.23% (+0.28)	95.77% (-0.18)	95.80% (-0.15)	+0.43%	0.72	0.71
OC	4103	160	336	160	43.4	336	160	336	88.03%	94.41% (+6.38)	91.65% (+3.62)	92.33% (+4.30)	+2.08%	1.00	1.00
HPD	4700	481	697	481	12.7	697	481	697	44.07%	68.13% (+24.06)	66.73% (+22.66)	66.30% (+22.23)	+1.40%	0.70	0.72
FLD	6864	502	970	502	34.9	970	502	970	44.99%	75.23% (+30.24)	72.02% (+27.03)	73.83% (+28.84)	+1.40%	0.70	0.60
TSR	9775	704	1860	704	53.2	1860	704	1860	81.65%	93.03% (+11.38)	92.63% (+10.98)	92.73% (+11.08)	+0.30%	0.83	0.67
OD	13194	258	770	258	69.1	770	258	770	84.12%	97.04% (+12.92)	96.63% (+12.51)	96.67% (+12.55)	+0.37%	0.79	0.60

IS: Improvement Set, US: Unsafe Set, BLUS: Balanced Labeled Unsafe Set; LUS: Labeled Unsafe Set (average over all the runs); ALUS/ALIS: Augmented Labeled Unsafe/Improvement Set.

We consider two baseline approaches, namely B1 and B2. B1 has been introduced in Section II-B and consists of selecting for retraining the misclassified images belonging to the labeled improvement set. B2 is depicted in Figure 13. It follows the HUDD process except that it selects unsafe images randomly (i.e., the *Reduced improvement set*) instead of relying on root cause clusters. B2 enables the evaluation of the benefits of selection based on root cause clusters over random selection.

To not introduce bias in the results, we rely on the same experiment setting for all the approaches (i.e., same configuration of the DNN training algorithm and same number of images to be labeled). In the case of HUDD, only the images in the unsafe set need to labeled. In the other cases, all the images in the improvement set must be labeled. For this reason, for the two baselines, we select an improvement set that is a random subset of the improvement set used by HUDD (referred to as *reduced improvement set*) and has the same size as the unsafe set generated by HUDD. To account for randomness, we repeat the experiment 10 times.

With HUDD, retraining the DNN was done by applying the approach described in Section IV-D. For B1 and B2, we configure bootstrap resampling to generate an *augmented labeled unsafe set* and an *augmented labeled improvement set* with the same size as the *balanced labeled unsafe set* for HUDD.

To answer the research question, we compute the accuracy of the retrained models on the test set and compare the accuracy improvement obtained by HUDD with that obtained by the baselines. We considered all the case studies listed in Table I. The improvement set for GD and OC has been generated through additional executions of UnityEyes. To simulate a realistic scenario in which engineers collect additional data from the field or construct additional simulator models to improve DNN accuracy, the improvement sets for HPD and FLD have been generated with additional executions of the IEE simulator configured to use two new face models, which were not used for generating the training and test sets. For the other cases, we selected images of the original datasets which had not been used for the training and test sets.

Results. The first eight columns of Table V provide the number of images used to retrain the DNNs. The remaining columns of Table V show the accuracy of the retrained models, the delta with respect to the best baseline, and $\hat{A}_{12}$ effect size [50]. For the accuracy, negative values indicate that the accuracy of the retrained model is worse than that of the original model. HUDD always fares better than the baseline approaches. Vargha and Delaney's $\hat{A}_{12}$ effect size is always equal or above 0.60, which indicates that, in all cases, HUDD has higher chances of generating accurate DNNs than baselines [50], [51]. In the paper by Vargha and Delaney [50], the authors specify the an effect size above 0.56 suggests a significant difference, with higher thresholds for medium (0.64) and large (0.71) effects.

HUDD accuracy improvements range from 0.28% to 30.24%. In contrast, B1 and B2 improvements range from -0.18% to 27.03% and -0.15% to 28.84%, respectively. For DNNs with an accuracy above 80%, we have the following ranges: from 0.28% to 12.92% (HUDD), from -0.18% to 12.51% (B1), and -0.15% to 12.55% (B2). For HPD and FLD, which have lower initial accuracy, we have the following ranges: from 12.92% to 24.06% (HUDD), from -0.18% to 12.51% (B1), and -0.15% to 12.55% (B2). We can therefore conclude that HUDD is most useful when DNNs have lower accuracy and there is more room for improvement. Those are also the cases where retraining is most particularly important. The negative results obtained by the baselines for GD suggest that retraining the DNN without targeting the DNN-error root causes may lead to worse accuracy. The choice of an inadequate strategy for retraining DNNs is therefore particularly detrimental since one could invest significant time and effort in labeling improvement set images without getting any benefit.

The difference in accuracy improvement between HUDD and the best baseline ranges between 0.30 (TSR) and 2.08 (OC). Given that all techniques cost the same according to our experiment design, it is therefore recommended to use HUDD. In addition, there is a larger average difference ($\geq$ 1.40) between HUDD and baselines in the cases of OC, HPD, and FLD. There are three plausible reasons to explain these differences. One is that, for HPD and FLD, there is significant room for accuracy improvement in the original models. Second, to increase the accuracy of HPD and FLD, we require improvement set images that are very different from the training set ones (i.e., pictures generated with new face models). We deem this to be a realistic situation that generalizes beyond our case studies; for example, it might be observed also in autonomous driving systems where certain types of vehicles (e.g., e-scooters) are missing from the training set. Third, for the three DNNs above, the training set is missing unsafe situations where the predictions are expected to be challenging (e.g., very dim light in the image). The type of retraining described above is particularly important in our

TABLE VI: Experiments Execution Time

DNN	Training	Testing	HUDD Step 1	HUDD Steps 4-7	BL1	BL2	Testing Improved DNNs	Total
GD	2.66%	2.00%	3.82%	26.91%	31.97%	26.64%	5.99%	375.3
OC	1.34%	8.93%	18.30%	15.63%	15.63%	13.39%	26.79%	3.7
HPD	2.01%	2.41%	21.16%	22.93%	24.14%	20.11%	7.24%	20.7
FLD	2.81%	1.41%	2.81%	28.43%	35.60%	28.10%	0.84%	177.9
OD	2.80%	0.73%	6.83%	29.13%	30.81%	28.01%	1.68%	29.8
TSR	1.89%	1.94%	29.36%	20.47%	22.63%	18.86%	4.85%	30.9

context since the reduction of the unsafe input space is a key objective of safety engineering practices for the automotive industry [9].

Though the above accuracy differences may appear small, they may nevertheless be important in the context of critical applications where every percentage point in improvement matters. Furthermore, one should recall that, when we are dealing with highly accurate DNNs, room for improvement is limited.

Finally, results with OD show that HUDD achieves better accuracy than MODE (97.04% vs 89% [11] after DNN retraining). These results show the potential of HUDD which, in addition to a higher accuracy than MODE, also provides root cause clusters.

C. Execution Time

Table VI provides details about the time required to perform our experiments. It reports on the total execution time and how it is distributed across the different steps of HUDD and the execution of the baseline approaches. In Table VI, columns 5 to 7 refer to the cumulative time over 10 repetitions, column 8 refers to the cumulative time required for testing the retrained models for HUDD, BL1, and BL2. Our experiments took between 3.7 (OC) and 375.3 hours (GD) across DNNs, which highlights the large endeavor entailed by repeating experiments ten times in order to be able to draw statistical conclusions. Execution time is driven by the size of the data sets (i.e., executions with DNNs trained and tested with larger data sets took more time). The time required by HUDD to improve the DNN (i.e., *Steps 4-7*)— which includes also the identification of unsafe images—is similar with the time required by baseline approaches (columns 6 and 7), thus showing that HUDD does not introduce delays in the retraining process. The time required to perform HUDD Step 1 is significant (between 40 minutes and 14 hours); however, Step 1 can be executed overnight and help reduce human effort to identify root cause clusters.

To be able to execute experiments for 638.3 hours, we parallelized the executions of experiments using the HPC cluster of the University of Luxembourg [52]. We relied on Intel Xeon Gold 6132 nodes (2.6 GHz with four Tesla V100 16G SXM2).

D. Threats to validity

We target DNNs performing image analysis in the perception layer of safety-critical systems. To address threats to external validity, for RQ2, we have considered DNNs performing classification of body parts and road objects, which are typical features in automotive systems. Further, one regression DNN was also analyzed, thus showing the applicability of the approach beyond classification; to this end, we considered a DNN representative of the typical task of landmark detection. Though four subject DNNs out of six implement tasks motivated by IEE business needs, they address problems that are quite common in the automotive industry (i.e., angle determination and landmarks detection). To strengthen generalizability, for two of these four case studies (i.e., GD and OC), we relied on a third-party simulator used in computer vision research (i.e., UnityEyes). Further, we considered two case studies from related work (i.e., TSR and OD) that rely on real images. Our benchmark DNNs are therefore both diverse and representative.

For RQ1, we could only consider a subset of the case studies having high-resolution simulators available. Simulation is based on Blender [14], a readily available and widely used technology, thus making our results more representative. In our experiments, to objectively and systematically evaluate the quality of the generated clusters, we relied on the analysis of simulator parameters rather than a user study, which, however, should be undertaken in the future. Though the fidelity of simulator images is always a question, our experimental results do not show different trends for real and simulated images with respect to the number of clusters and accuracy improvements. Indeed, the number of root cause clusters identified for the two DNNs working with real-world images (OD and TSR), which are 14 and 20, are similar to the ones observed in DNNs with simulator images (ranging from 11 to 17). Also, the accuracy improvement obtained for OD and TSR, i.e., +11.38 and +11.92 (see Table V), is within the range observed with simulator images (i.e., +0.28 and +30.24).

In our work, we do not rely on the popular K-means algorithm because of its higher computational cost in our context (see Section III-B). However, more computation might be justified by better performance results, which may include a higher variance reduction in root cause clusters, a larger number of explanatory root cause clusters, and higher accuracy improvement. Since our experiments have shown that (1) HUDD generates cohese and explanatory root cause clusters (RQ1), (2) the room for accuracy improvement is small (RQ2 results show that, for four out of six DNNs, HUDD accuracy is above 93%), and (3) our experiments already took 638.3 hours to complete, the empirical comparison of HAC with K-means and other clustering algorithms has not been considered a priority in this work.

Though HUDD background technology (i.e., LRP and HAC) is context-independent, future work will investigate the evaluation of the approach in different contexts (e.g., space industry).

VI. Related Work

Most of the DNN testing and analysis approaches are summarized in recent surveys [53], [54]. However, research on the automated debugging and repair of DNNs is still at very early stages.

Under-approximation boxes [55] consist of the minimal set of neurons, belonging to a specific layer, that ensure a postcondition (e.g., the generation of a specific DNN output). When applied to explain misclassifications, they lead to heatmap-like images showing the minimal set of input pixels leading to the same DNN result. Similarly, Ribeiro et al., identify the image chunks that are sufficient to generate a certain DNN result [56]. Like heatmap generation techniques, these two approaches cannot automatically identify the root cause for a group of error-inducing images but require manual inspection for every error-inducing image.

Decision trees can identify patterns of neuron activations common to a same output class [55]. They are not used to explain misclassifications [55] since they cannot be applied to look for patterns in root cause clusters which are not known a priori.

MODE automatically identifies the images to be used to retrain a DNN [11]. However, it cannot identify the root causes of DNN errors, which is a major limitation in our context. HUDD and MODE differ also regarding the selection of images to be used for retraining, which, in the case of MODE, is not based on heatmaps but on training additional DNN layers that capture commonalities among neuron activations leading to DNN errors. MODE, therefore, entails repeated modification and retraining of the DNN under test, just to select the improvement set, which is a very expensive endeavor.

Surprise adequacy measures the degree of variation in neuron activations between a new image and the training images belonging to the same class [57]. Empirical results show that a retraining set with a varying degree of surprise adequacy improves DNN robustness against adversarial examples. However, it has never been adopted to improve accuracy for non-adversarial inputs. Also, like previous techniques, it cannot be used to identify root causes of DNN errors.

DeepFault identifies a set of suspicious neurons to synthesize new, adversarial images and improve DNN adversarial robustness [58]. Since it relies on synthesized adversarial inputs, it cannot improve accuracy for unsafe, non-adversarial inputs. Once again, it does not distinguish different root causes.

Apricot [59] repairs DNNs by changing the weights of the DNN model. It works by training multiple DNNs on subsets of the training and test sets. The repair process aims to minimize (maximize) the distance between the weights of the DNN to repair and the weights of DNNs leading to better (worse) accuracy. Unfortunately, the accuracy improvement achieved by Apricot is lower than 2%.

Gao et al. [60] and Engstrom et al [61] rely on image transformations (e.g., rotations) to augment the training set and improve DNN robustness, thus addressing a different problem.

Active learning had been proposed to minimize the number of inputs that require to be labeled to train a machine learning model [62]. State-of-the-art approaches identify the inputs for which an incrementally trained model generates uncertain results; for binary classification, uncertainty can be measured by means of entropy [62]. However, traditional active learning approaches that identify individual images for incremental retraining are unsuitable for CNNs that require a large training sets. For CNNs, a state-of-the-art approach consists of the generation of a coreset (a small summary of large data sets) with k elements [63]. It outperforms approaches based on uncertainty sampling [64]. Though promising, such approach addresses a problem different than ours, i.e., minimizing the cost of training, under the assumption that labelling cost is uniform for all the images. In our context, thanks to the use of simulators, labelling costs concern mostly the real-world images used to test or improve the accuracy of the DNN; for this reason, HUDD focuses on the selection of images to be used for re-training. Instead, coresets are not meant to be used to reduce an improvement set in the presence of a model that has already been trained on a large set of inputs. Another limitation of state-of-the-art active learning solutions is that they are inapplicable with regression DNNs. Indeed, solutions for regression models incur computational costs that are cubic with respect to the number of model parameters, thus being infeasible for the DNNs used in image processing tasks [65].

Other approaches aim to reduce the costs associated with the labeling of the test set by minimizing its size [66], [67]. Though they address a different problem, Chen et al. [67] rely on clustering applied to DNN features (e.g., neuron activations). More precisely, they use HDBSCAN [68] to identify an optimal configuration for the state-of-the-art DBSCAN algorithm [69]. Also, they demonstrate that the FastICA dimensionality reduction algorithm [70] is enabling HDBSCAN to produce the best clustering results. Though the authors point to the execution time of HDBSCAN as one of its main limitations, the combination FastICA and HDBSCAN should be evaluated as part of future work to improve the root cause clusters generated by HUDD.

With respect to a recent taxonomy of DNN faults [71], HUDD can identify different types of *training* and *input* faults, while automatically addressing problems due to *training data quality* (see Section IV-B). A more extensive evaluation of HUDD based on this taxonomy is part of our future work.

To summarize, HUDD is the first approach that facilitates the scalable identification of distinct failure root causes in DNNs, by applying clustering algorithms to heatmaps generated by DNN explanation techniques. For the latter, we rely on Layer-Wise Relevance Propagation (LRP), which is based on theoretical foundations that are generalizable to other DNN architectures. Further, HUDD relies on standard retraining procedures based on back propagation and gradient analysis, that have been widely applied and validated, and does not entail the direct modification of the learned DNN model. As demonstrated in our empirical evaluation, HUDD can successfully support the debugging of DNNs that implement either classification or regression tasks.

VII. Conclusion

In this paper we introduced HUDD, an approach that automatically identifies the different situations in which an image processing DNN is likely to produce erroneous results. HUDD generates clusters (i.e., root cause clusters) containing misclassified input images sharing a common set of characteristics that are plausible causes for errors. This is achieved through an hierarchical agglomerative clustering algorithm applied to heatmaps capturing the relevance of neurons across different DNN layers on the result.

In addition, HUDD minimizes the effort required to select and label additional images to be used to augment the training set and improve the DNN. This is done by automatically selecting images that are close to members of root cause clusters and thus unsafe. Only these selected images then need to be labeled by engineers. Since DNN errors are often due to an incomplete training set (e.g., lack of images with a gaze angle close to borderline), HUDD alleviates the problem by augmenting the training set with unsafe images.

Empirical evaluation with simulator images show that HUDD generates clusters of images sharing similar values for some of the simulation parameters driving the generation of images. We can conclude that such clusters can then serve as a useful instrument for the identification of root causes of DNN errors, as exemplified in our case studies. In turn, this information is important to safety analysis as it helps clearly characterize unsafe inputs, a requirement in safety standards. Our results, on both simulated and real images, also show how these clusters can be effectively used to select new images for retraining in a way that is more efficient than existing practices and leading to better DNN accuracy.

Acknowledgment

This project has received funding from IEE Luxembourg, Luxembourg's National Research Fund (FNR) under grant BRIDGES2020/IS/14711346/FUNTASY, the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 694277), and NSERC of Canada under the Discovery and CRC programs. Authors would like to thank Thomas Stifter and Jun Wang from IEE for their valuable support.

The experiments presented in this paper were carried out using the HPC facilities of the University of Luxembourg (see http://hpc.uni.lu).

References

[1] NVIDIA Corporation, "NVIDIA Introduces DRIVE AutoPilot," 2019. [Online]. Available: https://nvidianews.nvidia.com/news/nvidia-introduces-drive-autopilot-worlds-first-commercially-available-level-2-automated-driving-system

[2] Tesla, Inc., ""overview of neural net for vision, sonar and radar processing software"," 2019. [Online]. Available: https://www.tesla.com/BLOG/ALL-TESLA-CARS-BEING-PRODUCED-NOW-HAVE-FULL/-SELF-DRIVING-HARDWARE?redirect=no

[3] IEE, "IEE sensing solutions. www.iee.lu," 2020.

[4] ZF, Inc., ""dream safety"," 2019. [Online]. Available: https://www.zf.com/site/magazine/en/articles_3392.html

[5] R. A. Naqvi, M. Arsalan, G. Batchuluun, H. S. Yoon, and K. R. Park, "Deep learning-based gaze detection system for automobile drivers using a nir camera sensor," *Sensors*, vol. 18, no. 2, 2018.

[6] International Organization for Standardization, "ISO, ISO-24765-2017, Systems and software engineering - Vocabulary," 2020.

[7] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 618–626.

[8] International Organization for Standardization, "ISO, ISO26262-1:2018, Road vehicles: Functional safety," 2020.

[9] ——, "ISO/PAS 21448:2019, Road vehicles: Safety of the intended functionality," 2020.

[10] G. Montavon, A. Binder, S. Lapuschkin, W. Samek, and K. R. Müller, *Layer-Wise Relevance Propagation: An Overview*. Cham: Springer International Publishing, 2019, pp. 193–209. [Online]. Available: https://doi.org/10.1007/978-3-030-28954-6_10

[11] S. Ma, Y. Liu, W.-C. Lee, X. Zhang, and A. Grama, "Mode: Automated neural network model debugging via state differential analysis and input selection," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2018. New York, NY, USA: ACM, 2018, pp. 175–186.

[12] R. S. King, *Cluster Analysis and Data Mining: An Introduction*. USA: Mercury Learning & Information, 2014.

[13] E. Wood, T. Baltrušaitis, L.-P. Morency, P. Robinson, and A. Bulling, "Learning an appearance-based gaze estimator from one million synthesised images," in *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, ser. ETRA '16. New York, NY, USA: ACM, 2016, pp. 131–138.

[14] Blender, "Blender 3D simulation and rendering engine," 2020. [Online]. Available: https://www.blender.org/

[15] H. D. III, *A Course in Machine Learning*. Online Available, 2020. [Online]. Available: http://ciml.info/

[16] R. Garcia, A. C. Telea, B. C. da Silva, J. Torresen, and J. L. D. Comba, "A task-and-technique centered survey on visual analytics for deep learning model engineering," *Computers and Graphics*, vol. 77, pp. 30 – 49, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0097849318301535

[17] V. Petsiuk, A. Das, and K. Saenko, "Rise: Randomized input sampling for explanation of black-box models," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2018.

[18] P. Dabkowski and Y. Gal, "Real time image saliency for black box classifiers," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS?17. Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 6970–6979.

[19] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 818–833.

[20] J. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," in *ICLR (workshop track)*, 2015.

[21] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 2921–2929.

[22] G. Castanon and J. Byrne, "Visualizing and quantifying discriminative features for face recognition," in *2018 13th IEEE International Conference on Automatic Face Gesture Recognition (FG 2018)*, May 2018, pp. 16–23.

[23] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K. Muller, "Evaluating the visualization of what a deep neural network has learned," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 11, pp. 2660–2673, Nov 2017.

[24] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Muller, "Explaining nonlinear classification decisions with deep taylor decomposition," *Pattern Recognition*, vol. 65, pp. 211 – 222, 2017.

[25] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[26] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 483–499.

[27] G. Montavon, "WIFS 2017 Tutorial on Methods for Understanding DNNs and their Predictions," 2019. [Online]. Available: http://heatmapping.org/wifs2017/

[28] J. Ward, J. H., "Hierarchical grouping to optimize an objective function," *American Statistical Association Journal*, vol. 58, pp. 236–244, 1963.

[29] R. R. Sokal and C. D. Michener, "A statistical method for evaluating systematic relationships," *University of Kansas Science Bulletin*, vol. 38, pp. 1409–1438, 1958.

[30] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: an overview," *WIREs Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 86–97, 2012. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.53

[31] V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan, "Finding a "kneedle" in a haystack: Detecting knee points in system behavior," in *2011 31st*

International Conference on Distributed Computing Systems Workshops, 2011, pp. 166–171.

[32] Z. Li, C. Luo, Y. Zhao, Y. Sun, K. Sui, X. Wang, D. Liu, X. Jin, Q. Wang, and D. Pei, "Generic and robust localization of multi-dimensional root causes," in *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*, 2019, pp. 47–57.

[33] L. Jendele, M. Schwenk, D. Cremarenco, I. Janicijevic, and M. Rybalkin, "Efficient automated decomposition of build targets at large-scale," in *2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST)*, 2019, pp. 457–464.

[34] R. L. Thorndike, "Who belongs in the family?" *Psychometrika*, vol. 18, no. 4, pp. 267–276, 1953. [Online]. Available: https://doi.org/10.1007/BF02289263

[35] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability - Vol. 1*, L. M. Le Cam and J. Neyman, Eds. University of California Press, Berkeley, CA, USA, 1967, pp. 281–297.

[36] F. Murtagh and P. Legendre, "Ward's hierarchical agglomerative clustering method: Which algorithms implement ward's criterion?" *Journal of Classification*, vol. 31, no. 3, pp. 274–295, Oct 2014. [Online]. Available: https://doi.org/10.1007/s00357-014-9161-z

[37] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008. [Online]. Available: http://nlp.stanford.edu/IR-book/information-retrieval-book.html

[38] M. Inaba, N. Katoh, and H. Imai, "Variance-based k-clustering algorithms by voronoi diagrams and randomization," *IEICE Transactions on Information and Systems*, vol. 83, pp. 1199–1206, 2000.

[39] A. Vattani, "k-means Requires Exponentially Many Iterations Even in the Plane," *Discrete & Computational Geometry*, vol. 45, no. 4, pp. 596–616, 2011. [Online]. Available: https://doi.org/10.1007/s00454-011-9340-1

[40] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.

[41] B. Fornberg, "Generation of finite difference formulas on arbitrarily spaced grids," *Math. Comp.*, no. 51, pp. 699–706, 1988.

[42] S. Bakhshi, D. A. Shamma, L. Kennedy, Y. Song, P. de Juan, and J. J. Kaye, "Fast, cheap, and good: Why animated gifs engage us," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ser. CHI '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 575–586.

[43] PyTorch, "PyTorch DNN framework," 2020. [Online]. Available: https://pytorch.org

[44] SciPy, "Pyton framework for mathematics, science, and engineering." 2020. [Online]. Available: https://scipy.org/

[45] Authors of this paper, "HUDD: toolset and replicability package," 2020. [Online]. Available: https://sntsvv.github.io/HUDD/

[46] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[47] M. community, "MakeHuman computer graphics middleware for the prototyping of humanoids. ," 2020. [Online]. Available: http://www.makehumancommunity.org

[48] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 3730–3738.

[49] INI, "TRaffic Sign Dataset," 2020. [Online]. Available: http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset

[50] A. Vargha and H. D. Delaney, "A critique and improvement of the cl common language effect size statistics of mcgraw and wong," *Journal of Educational and Behavioral Statistics*, vol. 25, no. 2, pp. 101–132, 2000.

[51] 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: ACM, 2011, pp. 1–10.

[52] S. Varrette, P. Bouvry, H. Cartiaux, and F. Georgatos, "Management of an academic hpc cluster: The ul experience," in *Proc. of the 2014 Intl. Conf. on High Performance Computing & Simulation (HPCS 2014)*. Bologna, Italy: IEEE, July 2014, pp. 959–967.

[53] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, and X. Yi, "A survey of safety and trustworthiness of deep neural networks," 2018.

[54] J. M. Zhang, M. Harman, L. Ma, and Y. Liu, "Machine learning testing: Survey, landscapes and horizons," 2019.

[55] D. Gopinath, H. Converse, C. Pasareanu, and A. Taly, "Property inference for deep neural networks," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2019, pp. 797–809.

[56] M. T. Ribeiro, S. Singh, and C. Guestrin, "Anchors: High-precision model-agnostic explanations," in *Proceedings of the '18 AAAI Conference on Artificial Intelligence*, 2018. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16982

[57] J. Kim, R. Feldt, and S. Yoo, "Guiding deep learning system testing using surprise adequacy," in *Proceedings of the 41st International Conference on Software Engineering*, ser. ICSE '19. IEEE Press, 2019, pp. 1039–1049.

[58] H. F. Eniser, S. Gerasimou, and A. Sen, "Deepfault: Fault localization for deep neural networks," in *Fundamental Approaches to Software Engineering*, R. Hähnle and W. van der Aalst, Eds. Cham: Springer International Publishing, 2019, pp. 171–191.

[59] H. Zhang and W. K. Chan, "Apricot: A weight-adaptation approach to fixing deep learning models," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Nov 2019, pp. 376–387.

[60] X. Gao, R. K. Saha, M. R. Prasad, and A. Roychoudhury, "Fuzz testing based data augmentation to improve robustness of deep neural networks," in *Proceedings of the 42nd International Conference on Software Engineering*, ser. ICSE '20. New York, NY, USA: ACM, 2020.

[61] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry, "Exploring the landscape of spatial robustness," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. Long Beach, California, USA: PMLRG, 09–15 Jun 2019, pp. 1802–1811.

[62] B. Settles, "Active learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 18, no. 1, pp. 1–111, 2012. [Online]. Available: https://www.morganclaypool.com/doi/abs/10.2200/S00429ED1V01Y201207AIM018?ai=1ge{\&}mi=6a5dzh{\&}af=R

[63] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A core-set approach," in *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018, pp. 1–13.

[64] Y. Shen, H. Yun, Z. Lipton, Y. Kronrod, and A. Anandkumar, "Deep active learning for named entity recognition," in *Proceedings of the 2nd Workshop on Representation Learning for NLP*. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 252–256. [Online]. Available: https://www.aclweb.org/anthology/W17-2630

[65] B. Settles, *Active Learning : Active Learning*. San Rafael, UNITED STATES: Morgan & Claypool Publishers, 2011.

[66] Z. Li, X. Ma, C. Xu, C. Cao, J. Xu, and J. Lü, "Boosting operational dnn testing efficiency through conditioning," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2019. New York, NY, USA: Association for Computing Machinery, 2019, p. 499–509.

[67] J. Chen, Z. Wu, Z. Wang, H. You, L. Zhang, and M. Yan, "Practical accuracy estimation for efficient deep neural network testing," *ACM Trans. Softw. Eng. Methodol.*, vol. 29, no. 4, Oct. 2020.

[68] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering," *Journal of Open Source Software*, vol. 2, no. 11, p. 205, 2017. [Online]. Available: https://doi.org/10.21105/joss.00205

[69] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96. AAAI Press, 1996, p. 226–231.

[70] E. Oja and Z. Yuan, "The fastica algorithm revisited: Convergence analysis," *Trans. Neur. Netw.*, vol. 17, no. 6, p. 1370–1381, Nov. 2006. [Online]. Available: https://doi-org.proxy.bnl.lu/10.1109/TNN.2006.880980

[71] N. Humbatova, G. Jahangirova, G. Bavota, V. Riccio, A. Stocco, and P. Tonella, "Taxonomy of real faults in deep learning systems," in *Proceedings of the 42nd International Conference on Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2020.