

# Black-box Safety Analysis and Retraining of DNNs based on Feature Extraction and Clustering

MOHAMMED OUALID ATTAOUI, SnT Centre, University of Luxembourg, Luxembourg

HAZEM FAHMY, SnT Centre, University of Luxembourg, Luxembourg

FABRIZIO PASTORE, SnT Centre, University of Luxembourg, Luxembourg

LIONEL BRIAND, SnT Centre, University of Luxembourg, Luxembourg and School of EECS, University of Ottawa, Canada

Deep neural networks (DNNs) have demonstrated superior performance over classical machine learning to support many features in safety-critical systems. Although DNNs are now widely used in such systems (e.g., self driving cars), there is limited progress regarding automated support for functional safety analysis in DNN-based systems. For example, the identification of root causes of errors, to enable both risk analysis and DNN retraining, remains an open problem. In this paper, we propose SAFE, a black-box approach to automatically characterize the root causes of DNN errors. SAFE relies on a transfer learning model pre-trained on ImageNet to extract the features from error-inducing images. It then applies a density-based clustering algorithm to detect arbitrary shaped clusters of images modeling plausible causes of error. Last, clusters are used to effectively retrain and improve the DNN. The black-box nature of SAFE is motivated by our objective not to require changes or even access to the DNN internals to facilitate adoption.

Experimental results show the superior ability of SAFE in identifying different root causes of DNN errors based on case studies in the automotive domain. It also yields significant improvements in DNN accuracy after retraining, while saving significant execution time and memory when compared to alternatives.

CCS Concepts: • **Software and its engineering** → **Software defect analysis**; • **Computing methodologies** → **Machine learning**.

Additional Key Words and Phrases: DNN Explanation, DNN Functional Safety Analysis, DNN Debugging, Clustering, Transfer Learning

## ACM Reference Format:

Mohammed Oualid Attaoui, Hazem Fahmy, Fabrizio Pastore, and Lionel Briand. 2022. Black-box Safety Analysis and Retraining of DNNs based on Feature Extraction and Clustering. 1, 1 (October 2022), 41 pages. <https://doi.org/10.1145/nnnnnnnn.nnnnnnn>

## 1 INTRODUCTION

Deep neural networks (DNN) have become an essential computational tool inside many cyber-physical systems. This success is partly due to their capacity to automate complex tasks that are typically performed by humans and are difficult to program. It is also driven by the high performance

---

Authors' addresses: Mohammed Oualid Attaoui, SnT Centre, University of Luxembourg, JFK 29, Luxembourg, Luxembourg, [mohammed.attaoui@uni.lu](mailto:mohammed.attaoui@uni.lu); Hazem Fahmy, SnT Centre, University of Luxembourg, JFK 29, Luxembourg, Luxembourg, [hazem.fahmy@uni.lu](mailto:hazem.fahmy@uni.lu); Fabrizio Pastore, SnT Centre, University of Luxembourg, JFK 29, Luxembourg, Luxembourg, [fabrizio.pastore@uni.lu](mailto:fabrizio.pastore@uni.lu); Lionel Briand, SnT Centre, University of Luxembourg, JFK 29, Luxembourg, Luxembourg, School of EECS, University of Ottawa, Ottawa, Canada, [lionel.briand@uni.lu](mailto:lionel.briand@uni.lu).

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

© 2022 Association for Computing Machinery.

XXXX-XXXX/2022/10-ART \$15.00

<https://doi.org/10.1145/nnnnnnnn.nnnnnnn>

they have achieved in many important fields regarding perception and decision-making tasks in smart grids [32, 77], networked surveillance [14, 72], medical imaging [10, 58], and autonomous vehicles [39, 71]. A good example of the latter is IEE [27], our industry partner in this research, who is extending its portfolio of in-vehicle monitoring systems with DNN-based products.

A DNN model is often regarded as a black-box. Despite their high performance, such models cannot easily provide meaningful explanations on how a specific prediction (decision) is made. Without such explanations to enhance the transparency of DNN models, it remains challenging to build up trust and credibility among end-users, especially in the context of safety-critical systems that need to be certified.

Such trustworthiness, for a DNN model, can be addressed predominantly by two processes: a certification process and an explanation process [24]. The certification process is held before the deployment of the product to ensure that it is reliable and safe. The explanation process is performed whenever needed during the lifetime of the product. Explanation is required in the safety-critical context to support safety analysis. The safety standards, such as ISO26262 [30], and ISO/PAS 21448 [29], 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).

Explanation methods aim at making neural networks decisions trustworthy [16]. It is usually defined as a visual aid accompanying a prediction to provide insights into the underlying reasons for the model output. Existing works in the literature have provided alternative techniques to explain DNNs [31], which focus on different model elements, e.g., the training dataset or the learned feature representations.

When DNN-based systems are used in a safety-critical context, root cause analysis is required to support safety analysis [12]. A root cause is a source of a failure, which is in our context an incorrect DNN prediction or classification.

One example root cause may be that certain classes are harder to distinguish. For example, in CIFAR-10 [35], dog and cat classes tend to confuse the DNN model since they share many standard semantic features. For multi-label classification, one root cause may be that two classes frequently appear together. For example, in the COCO dataset [40], mouse and laptop appear in the same image frequently, making it hard for the DNN model to distinguish between them [70].

In general, there are two categories of root cause analysis methods based on machine learning [17]: supervised and unsupervised. Supervised methods perform well in the systems where the different classes of problems are known a priori. Several supervised learning methods have been used for root cause analysis, for instance, SVM [22] and Bayesian models [2].

Unlike supervised methods, unsupervised methods do not require any training labels but automatically cluster failures and mine common features in each cluster. Several unsupervised root cause analysis approaches have been proposed in the literature, for instance, Sparse Filtering [38], and Frequent Pattern Mining [51].

Though supervised root-cause analysis is widely used, it is not adequate for all scenarios, especially when labels are missing or not numerous enough. Further, in modern architectures, engineers cannot guess all potential failure causes. Moreover, new root causes can appear when changing configurations and settings. Thus, more dynamic and less human-dependent, unsupervised root-cause-analysis methods have been proposed [12, 17, 51]. These unsupervised methods automatically cluster failures according to common causes, without expert's involvement. The main limitation of this approach is in the clustering evaluation. In this step, the authors use external validation measures to evaluate clustering quality. Such measures require the data to be labeled, for example Normalized Mutual Information [67] and Adjusted Rand Index [25]. However, in most case studies,

the ground truth is not available and we have no other choice but to use internal measures like the Silhouette Index [57] and the Davies Bouldin Index [8].

The current paper proposes SAFE (Safety Analysis based on Feature Extraction), a new automated approach for root cause analysis. The foremost objective is to provide a black-box solution that does not rely on internal information about the DNN or its modification, thus facilitating its adoption in practice. Indeed, engineers do not have access to such information in many contexts or are not sufficiently skilled to modify the DNN, whose development is often outsourced. Our approach targets DNNs that process images since they are the most common form of inputs for many DNN-based components in the automotive and other safety-critical domains (e.g., manufacturing robots). SAFE can be extended to deal with other types of inputs by choosing a feature extraction method adapted to this kind of data (e.g., BERT [9] for text data).

SAFE makes use of transfer learning-based feature extraction, dimensionality reduction, and unsupervised learning. This approach is an improvement over HUDD [12] to avoid reliance on heatmap-based distance, which requires access to the DNN's internal information, and to improve the quality of the root cause clusters' identification. Transfer Learning transfers the knowledge from a generic domain to another specific domain using a pre-trained model. Besides a large amount of time saved by using these methods, it has been shown that starting from a pre-trained model may perform better than training from scratch even on a different problem [43, 73]. In our approach, we propose to extract the features from our error-inducing images based on convolutional layers in a pre-trained model instead of relying on heatmaps.

We conducted an empirical evaluation on six DNNs. Our empirical results show the cost-effectiveness of SAFE in identifying plausible root causes with a reasonable human effort and its efficiency in memory usage and computation time. SAFE also achieved significant improvements in the retraining of DNNs (up to 35% improvement over the original models) and overall better results than alternatives (e.g., HUDD).

The rest of the paper is organized as follows. In Section 2, we present HUDD and its main features and limitations. In Section 3, we describe our proposed approach and its expected advantages. In Section 4, we present the experiment questions, design, and results, including a comparison with HUDD. In Section 5, we discuss and compare related work. Finally we conclude this paper in Section 6.

## 2 BACKGROUND

This Section introduces the body of work on which we build our approach, with a focus on DNN explanation and transfer learning-based feature extraction. We also describe our previous approach, HUDD [12] (Heatmap-based Unsupervised Debugging of DNNs), which is used as a baseline of comparison.

### 2.1 DNN Explanation and HUDD

Approaches that aim to explain DNN results have been developed in recent years [15]. Most of these concern the generation of heatmaps that capture the importance of pixels in image predictions. They include black-box [7, 53] and white-box approaches [48, 61, 65, 79, 85]. Black-box approaches generate heatmaps for the input layer and do not provide insights regarding internal DNN layers. White box approaches rely on the backpropagation of the relevance score computed by the DNN [48, 61, 65, 79, 85].

For example, Layer-Wise Relevance Propagation (LRP) [48] redistributes the relevance scores of neurons in a higher layer to those of the lower layer. Figure 1 illustrates the execution of LRP on a fully connected network used to classify inputs. 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. The heatmap in Figure 1 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. In his backward pass, LRP generates *internal heatmaps*. An internal heatmap for a DNN layer  $k$  consists of a matrix with the relevance scores computed for all the neurons of layer  $k$ .

Although heatmaps may provide useful information to determine the characteristics of an image that led to an erroneous result from the DNN, they are of limited applicability because, to determine the cause of all DNN errors observed in the test set, engineers may need to visually inspect all the error-inducing images, which is practically infeasible. To overcome such limitation, we recently developed HUDD [12], a technique that facilitates the explanation and removal of the DNN errors observed in a test set. HUDD generates clusters of images that led to a DNN error because of a same root cause. The root cause is determined by the engineer who visualizes a subset of the images belonging to each cluster and identifies the commonality across each image (e.g., for a Gaze detection DNN, all the images present a closed eye). To further support DNN debugging, HUDD

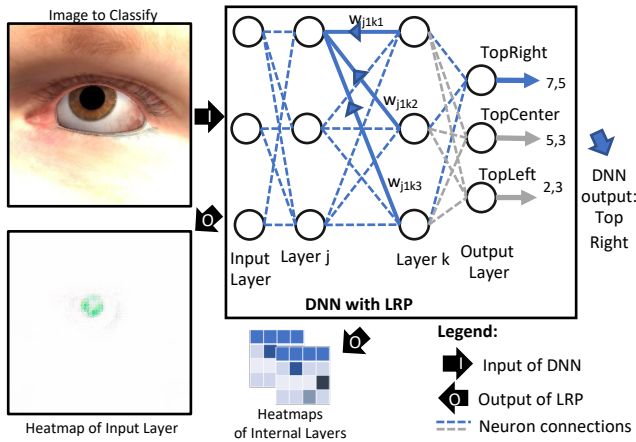


Fig. 1. Layer-Wise Relevance Propagation.

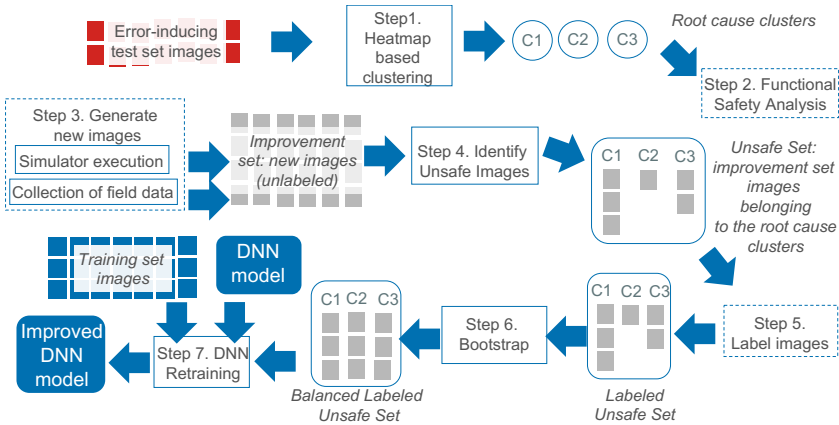


Fig. 2. Overview of HUDD.

automatically retrains the DNN by selecting from a pool of unlabeled images a subset that will likely lead to DNN errors because of the same root causes observed in the test set.

HUDD consists of seven steps, shown in Figure 2. In Step 1, HUDD performs heatmap-based clustering, which consists of three activities: (1) generate heatmaps for the error-inducing test set images, (2) compute distances between every pair of images using the euclidean distance applied to their heatmaps, and (3) execute hierarchical agglomerative clustering to group images based on the computed distances. Heatmaps enable HUDD to determine similarities based on a characteristic that actually caused the erroneous DNN result. 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, if needed.

In Step 3, engineers select a new set of unlabeled real-world 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.

In Step 5, engineers manually label the images belonging to the unsafe set. Different from traditional practice, 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 (i.e., replicating samples in the unsafe set) in order to have a sufficiently large number of unsafe images to improve the DNN.

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.

Although effective, HUDD presents a number of limitations. First, it can only analyze DNN implementations extended to compute LRP. Although LRP implementations for multiple DNN architectures relying on the tensorflow framework are available [4], it might be particularly complex for engineers to integrate LRP into a different DNN architecture. Indeed, the relevance computation formula to be adopted for each layer depends on the layer type (e.g., input, normalization, spatial pooling, internal layer) and the presence of recursion [48].

Also, companies often acquire off-the-shelf DNNs which cannot be modified, thus preventing the computation of LRP and the application of HUDD. Moreover, computing a heatmap-based euclidean distance might become particularly expensive when layers are made of thousands of neurons and hundreds of error-inducing images need to be processed. Finally, given that the neurons relevant for a specific DNN error (i.e., the neurons with high relevance scores) might represent a small proportion of the neurons in a DNN layer, computing the euclidean distance considering all the items in a heatmap may potentially lead to imprecise clusters caused by noise (i.e., the sum of many differences that are almost zero).

For all the reasons above, although the safety analysis and improvement of DNNs through the automated identification of root cause clusters has demonstrated to be effective, achieving a wider adoption requires a black-box substitute for HUDD.

## 2.2 Transfer Learning and Feature Extraction

To maximize the accuracy of DNNs in a cost-effective way, engineers often rely on the transfer learning approach, which consists of transferring knowledge from a generic domain, usually ImageNet [66], to another specific domain, (e.g., Safety Analysis, in our case). In other terms, we try to exploit what has been learned in one task and improve generalization in another task. Researchers have demonstrated the efficiency of transfer learning from ImageNet to other domains [69]. The hierarchical nature of convolutional neural networks (CNNs) encouraged the computer vision community to use this technique with distant datasets due to the similarities between the features extracted by the first CNN layers. Transfer learning saves training time, gives better performance in most cases, and reduces the need for a large dataset.

Transfer learning-based *Feature Extraction* is an efficient method to transform unstructured data into structured raw data to be exploited by any machine learning algorithm. In this method, the features are extracted based on a pre-trained CNN model [10].

The standard CNN architecture [3, 64, 83] comprises three types of layers: convolutional layers, pooling layers, and fully connected layers. The convolutional layer is considered the primary building block of a CNN. This layer extracts relevant features from input images during training. Convolutional and pooling layers are stacked to form a hierarchical feature extraction module. The model captures the resulting feature map from the last pooling layer as a 3D matrix of size  $(N, N, N_c)$ , where  $N$  is its width and height, and  $N_c$  is its depth. For features extraction, this feature map is flattened to form a vector of size  $(1, N \times N)$ . In summary, the CNN model receives an input image of size  $(224, 224, 3)$ . This image is then passed through the network's layers to generate a vector of features. The feature extraction process from all images generates raw data represented by a 2D matrix (denoted as  $X$ ) formalized below:

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1m} & l_2 \\ x_{21} & x_{22} & \dots & x_{2m} & l_c \\ \dots & \dots & \dots & \dots & \dots \\ x_{k1} & x_{k2} & \dots & x_{km} & l_1 \end{bmatrix}, l_i \in \{l_1, l_2, \dots, l_c\} \quad (1)$$

where  $l_i$  represent the class labels,  $c$  is the number of categories,  $m = N \times N$  is the number of features, and  $k$  is the size of the dataset. In our case, the class categories will not be used since our approach is unsupervised. They are useful if the user is working on a supervised problem or if fine-tuning is required.

There are several pre-trained models to extract features based on transfer learning: InceptionV3 [68], VGGNet [63], ResNet50 [21], and MobileNet [23]. The extracted features are related to the used architecture, where, in our context, InceptionV3, VGGNet-16, VGGNet-19, ResNet50, and MobileNet generated 2048, 512, 512, 2048, and 1024 features, respectively. We notice that the VGGNet architectures extract the least amount of features. We rely on VGGNet-16 instead of VGGNet-19 because the latter is more costly in execution time (19 layers instead of 16 for VGGNet-16). We describe in the following the VGGNet architecture.

VGGNet [63] is a CNN characterized by a high number of layers (11 to 19 layers). The purpose of this architecture is to minimize the number of trainable parameters. Controlling the number of parameters helps to reduce overfitting issues. To this end, VGGNet proposes to increase the network's depth and to decrease the size of filters from  $7 \times 7$  and  $5 \times 5$  to  $3 \times 3$ . The comparative study between the number of parameters in 3 stacked convolutional layers associated with  $3 \times 3$  filters and a single convolutional layer associated with  $7 \times 7$  filters demonstrated that small filters reduce the total number of parameters. Also, it enhances non-linearity through ReLu activation functions in intermediate layers.

VGGNet proposes six different configurations: A, A-LRN, B, C, D, and E, where the depth varies from 11 to 19 layers. In this contribution, we exploited the configurations D and E, which are composed of 16 and 19 layers, respectively.

### 3 THE SAFE APPROACH

In this Section, we present SAFE, a solution to overcome the previously mentioned limitations of HUDD. SAFE relies on a new black-box approach to extract features and compute root cause clusters. This black-box approach is based on transfer learning and dimensionality reduction. SAFE also allows the detection of non-convex clusters [34]. A cluster is convex if, for every pair of points within this cluster, every point on the straight line segment that joins them is also within the cluster [34]. This kind of clusters is usually represented by its centroid. But in many practical cases, the data leads to clusters with arbitrary, non-convex shapes. Such clusters, however, cannot be properly detected by a centroid-based algorithm or even a hierarchical clustering algorithm, as they are not designed for arbitrary-shaped clusters.

The presented method has the following merits compared to HUDD:

- It avoids the need for extending the DNN under analysis to compute LRP, which is achieved by relying on a transfer learning model that extracts features from the images.
- It applies a feature-based distance instead of a heatmap-based distance, thus saving training time and memory.
- It applies a density-based clustering algorithm to detect non-convex clusters, modeling the DNN errors' root causes.
- It relies on the non-convex root cause clusters to select an unsafe set for the retraining of the DNN.

Figure 3 presents an overview of SAFE. It consists of six steps. In Step 1, root cause clusters are identified by relying on feature extraction-based clustering. Step 2 involves a visual inspection performed by engineers as required by functional safety analysis [29, 30]. In Step 3, a new set of images, referred to as the *improvement set*, is provided by the engineers to retrain the model. In Step 4, SAFE automatically selects a subset of images from the improvement set called the *unsafe set*. The engineers label the images in the unsafe set in Step 5. Finally, in Step 6, SAFE automatically retrains the model to enhance its prediction accuracy.

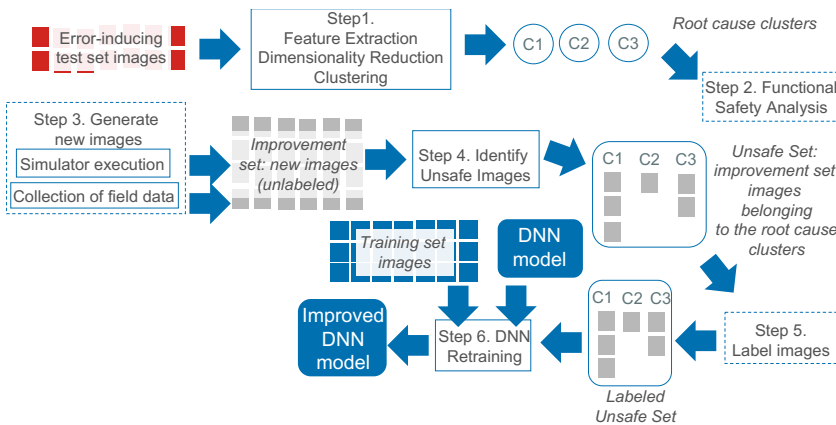


Fig. 3. Overview of SAFE

The main contributions of SAFE, when compared to HUDD, are in Step 1 and Step 4. Step 1 consists of four stages, namely (i) data acquisition and preprocessing, (ii) features extraction, (iii) dimensionality reduction, and (iv) clustering. Step 4 relies on a new method for the identification of the unsafe set that fits the clustering solution integrated in SAFE.

Similar to HUDD, SAFE includes some manual activities: Step 2 (visual inspection of images for functional safety analysis), Step 3 (generate new images), and Step 5 (label images). Such activities are, however, required also by state-of-the-practice approaches. Indeed, to debug and improve DNNs, engineers usually visually inspect all error-inducing images, select an improvement set, and manually label the images to be reused for retraining the DNN. However, both HUDD and SAFE significantly reduce the costs associated with these activities. Indeed, by inspecting a few representative images for each root cause clusters, instead of the whole set of error-inducing images, engineers can save substantial effort; further, since clusters group similar images together and each cluster is considered, it is less likely for engineers to overlook characteristics and associated causes appearing in a small subset of the images. Also, Step 2 is required only for functional safety analysis (e.g., to determine the unsafe cases to be discussed in safety analysis documents); it is not necessary if engineers aim at automated DNN improvement only. The effort required for the acquisition of the improvement set is the same for both HUDD, SAFE, and common practice (e.g., purchase an additional stock of real-world pictures); however, HUDD and SAFE require only the unsafe images to be labelled, thus reducing development costs. Finally, by relying on feature-extraction rather than heatmaps, SAFE eliminates the effort required to integrate LRP-based heatmap generation into the DNN under analysis, which is required by HUDD.

Figure 4 depicts the stages composing Step 1. We provide further details about each stage in the following subsections. Steps 2 and 4 are described last. Steps 3 and 5 are not further described because they are standard practice.

### 3.1 Step1.1: Data Preprocessing

This step aims to downsample the image sizes to the size required by the transfer learning models. Since we rely on a VGG16 model (see Section 3.2), which requires an input size of  $224 \times 224$  pixels, the images have to be resampled to match this requirement. To downsample the size of each image we rely on the Python Numpy reshape function<sup>1</sup>, which changes the shape of an array so that it

<sup>1</sup><https://sparrow.dev/numpy-reshape/>

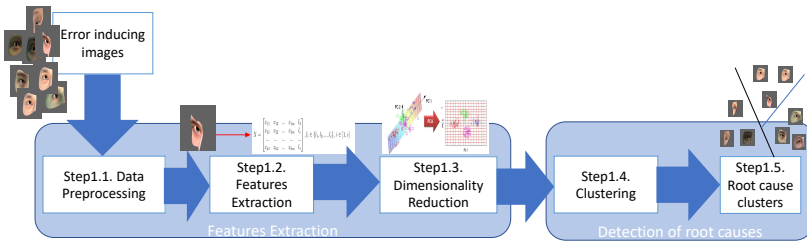


Fig. 4. Generation of root cause clusters with SAFE



has a new, compatible shape<sup>2</sup>. In our experiments, we downsampled images from  $(640 \times 480)$  to  $(224 \times 224)$ .

### 3.2 Step1.2: Feature extraction

Feature extraction aims to transform unstructured data into structured data for their exploitation by clustering algorithms [19].

As discussed in Section 2.2, we rely on transfer learning-based feature extraction. More precisely, we rely on VGGNet-16 models pre-trained on the ImageNet database.

### 3.3 Step1.3: Dimensionality reduction

Dimensionality reduction aims at approximating data in high-dimensional vector spaces [18].

This can be achieved using projections on hyperplanes. These methods, referred to as linear dimensionality reduction, include the well-known Principal Component Analysis (PCA) [52, 62]. Principal component analysis (PCA) is used for dimensionality reduction by projecting each data point onto the first few principal components, i.e., eigenvectors of the data covariance matrix, to obtain lower-dimensional data while preserving as much of the data variation as possible. The first principal component can equivalently be defined as a direction that maximizes the variance of the projected data [52, 62].

There are other methods to reduce dimensionality, including UMAP [47], LDA [78] and T-SNE [75]. We compared these methods with respect to clustering quality as measured by the Silhouette Index [57]. Two methods appeared to fare better, with comparable clustering quality: UMAP and PCA, the latter showing a much shorter execution time. In addition, PCA removes correlated features in contrast to UMAP.

### 3.4 Step1.4: Clustering

Clustering is an unsupervised learning technique that divides a set of objects into clusters: (a) objects in the same cluster must be as similar as possible, (b) objects in different clusters must be as different as possible.

Since the root cause clusters of images may have any shape and their number  $K$  cannot be determined a priori, we rely on an automatic  $K$ -determination clustering algorithm that can find clusters of arbitrary shapes. We use DBSCAN (density-based spatial clustering of applications with noise) [11], which is the most used density-based clustering algorithm [34, 46]. Intuitively, regions with high density are considered clusters and points with the most neighbors are referred to as the "core" of the clusters. The points with fewer neighbors are considered noise. The DBSCAN algorithm relies on two concepts: Reachability and Connectivity.

- **Reachability:** A point is reachable from another if the distance between them is inferior to a threshold  $\epsilon$ .
- **Connectivity:** If two points  $p$  and  $q$  are connected (i.e.,  $p$  is in the neighborhood of  $q$  based on  $\epsilon$ ) they belong to the same cluster.

DBSCAN introduces two parameters to apply these concepts:  $MinPts$  and  $\epsilon$ .

- **$MinPts$ :** The minimum number of points that a region (a hypersphere with a diameter equal to  $\epsilon$ ) should have to be considered dense.
- **Threshold  $\epsilon$ :** A threshold to determine if a point belongs to another point's neighborhood.

<sup>2</sup>The reshape function just changes the shape of the array containing the data (not the image itself) to match the VGG requirement. It does not change the data. The new shape must include the same total number of elements as the original shape.

SAFE uses a common technique to choose optimal values of  $\epsilon$  and *MinPts*. To select  $\epsilon$ , SAFE relies on the elbow method [6]. For this step, unlike that to find an optimal *MinPts*, SAFE does not require the execution of DBSCAN.

More specifically, the optimal value for  $\epsilon$  is selected as follows:

- First, we calculate the euclidean distance from each point to its closest neighbor.
- Then, we compute the average distance of every point to its closest neighbor and plot these distances in ascending order.
- Finally, we find the plot's elbow point [55], which is a point where there is a sharp change in the distance plot, which serves as a threshold. This point corresponds to the optimal  $\epsilon$  value.

To find the optimal value for *MinPts*, we run DBSCAN with  $\epsilon$  equal to the optimal value found above, and with different values for *MinPts*. We then select the clustering configuration with the highest Silhouette Index value [57]. The Silhouette index computes the compactness and the separateness of clusters. For a data point  $x_i$  assigned to cluster  $C_i$ , the Silhouette index is calculated as follow:

$$SI(i) = \frac{(b(i) - a(i))}{\text{Max}(b(i) - a(i))} \quad (2)$$

where  $a(i)$  is the average distance between  $x_i$  and all the data points assigned to cluster  $C_i$ .  $b(i)$  is the minimum average distance between  $x_i$  and the data points assigned to one of the other clusters  $C_j$  where  $j = 1, \dots, K; j \neq i$ . Based on the concepts described above, DBSCAN defines three types of points:

- Core point: It has at least *MinPts* points within a distance of  $\epsilon$ .
- Border point: It is not a core point, but it belongs to at least one cluster. That means that it lies within a distance  $\epsilon$  from a core point.
- Noise point: It is a point that is neither a core point nor a border point.

The DBSCAN algorithm proceeds by sampling points randomly from the dataset until all points are selected. For each point, it determines if it is a Core, Border or Noise point based on the parameters  $\epsilon$  and *MinPts*. The core points are considered representative of clusters. The clusters are then expanded by recursively repeating the neighborhood calculation for each point within the region. In DBSCAN, randomness affects only the selection of the point to be treated next; however, since each point ends up being assigned to the cluster containing the closest core point, randomness affects results only when border points are reachable from more than one cluster, which is unlikely [59]. If we run the algorithm several times with the same parameter settings, the same clusters will likely be obtained each time.

DBSCAN also helps identifying rare cases in the set of error-inducing images. Indeed, if there are enough rare cases to form a cluster (i.e., there are at least *MinPts* data points within a hypersphere with a diameter equal to  $\epsilon$ ), they are grouped together. If rare cases cannot form a cluster, most of them are considered noise and excluded from the set of clusters returned to the end-user. Since we select *MinPts* and  $\epsilon$  based on the analysis of the distribution of the error-inducing images, rare cases are considered noise only if they occur in regions of low density (i.e., the region contain less images than *MinPts*). Figure 5 shows an example of a cluster containing images representing a rare case (the eye is in an unusual position). In this case, there are ten error-inducing images with such characteristic. When *MinPts*  $\leq 10$ , then all the ten images form a cluster. However, if *MinPts*  $> 10$ , the images will be considered noise. In practice, *MinPts* is unlikely to be above 10 because its value is automatically derived considering all the dense regions, including the area with these rare cases. Further, if rare cases that are considered noise share similarities with images in other clusters, they will be assigned to the cluster with the most similar features. However, this case is unlikely to happen since such rare cases appear in sparse areas while clusters only appear in

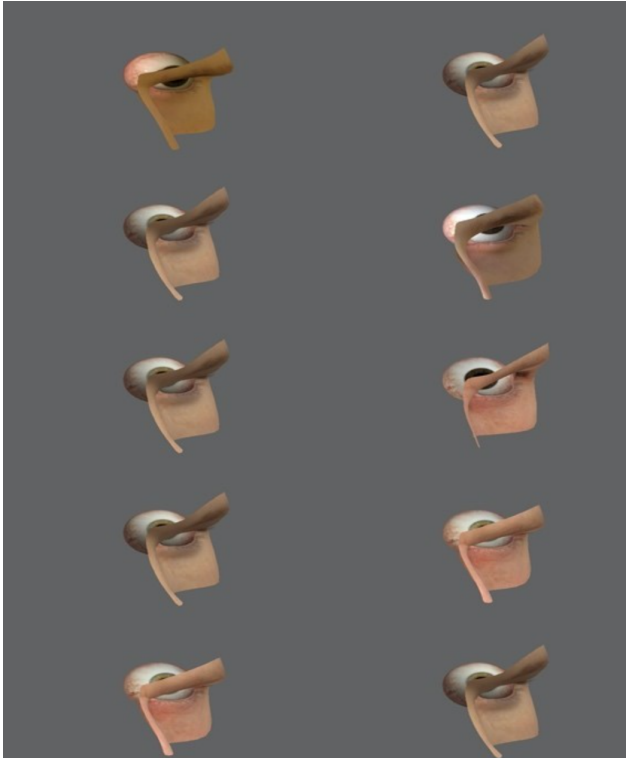


Fig. 5. Example of a root cause cluster with rare cases

dense areas. In practice, a rare case assigned to a cluster might actually share the same root cause of the other images belonging to the cluster.

### 3.5 Step1.5: Functional safety analysis through root cause clusters visualization

To analyze the clusters, safety engineers can use the same approach as in HUDD [12]. For each cluster, a subset of elements is visually inspected to determine the associated unsafe conditions, as functional safety analysis requires. Similarities among images within each cluster may suggest the cause of failures of the DNN. In other words, engineers attempt to identify the root cause of each cluster to gain a better understanding of the DNN behavior.

Figure 6 shows examples of root cause clusters identified by SAFE for the Head Pose Detection case study subject considered in our empirical evaluation (see Section 4). We notice that in cluster 1 the hidden eye seems to confuse the DNN and is a plausible reason for failure. The same observation can be made for Cluster 2, where, because the head is turned to the right, the right eye is not visible. Clusters 3 and 4 identify common causes of error due to an incomplete training set, i.e., the absence of images with a head pose close to a borderline.

In general, SAFE generates clusters that differ for at least one common characteristic in the images. For example, the root causes presented by Cluster 1 and Cluster 2 are not the same. Indeed, Cluster 1 concerns the right eye while Cluster 2 concerns the the left eye. Engineers might be interested in knowing if the training set is lacking images with only the left eye being hidden or both; for this reason, we believe that separating such clusters is beneficial. Empirical results are reported in Section 4.2.4.

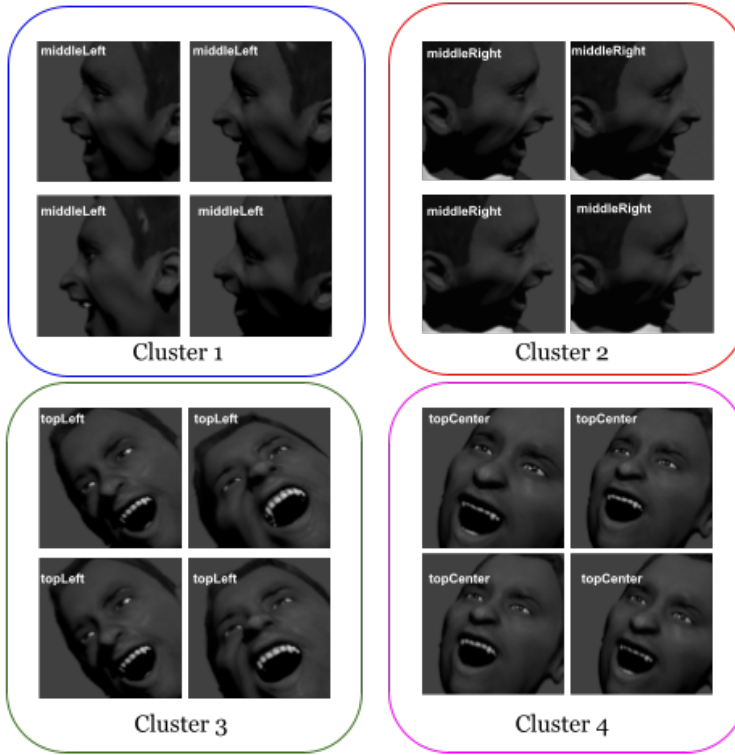


Fig. 6. Examples of root cause clusters for Head Pose detection

Similar to HUDD, SAFE can identify different root causes of errors, including (1) borderline cases (e.g., the gaze and head pose angle detected by Cluster 1 in Figure 6), (2) an incomplete training set (e.g., Clusters 3 and 4 in Figure 6), (3) an incomplete definition of the predicted classes (e.g., Cluster 1 in Figure 7 shows a cluster generated from our GD case study in Section 4.2.4, with eyes looking middle center, a class missing from our configuration) and (4) limitations in our capacity to control the simulator (e.g., unlikely face positions detected by Cluster 2 in Figure 7). In general, SAFE identifies commonalities (i.e., root causes) across images leading to failures. Based on a recent taxonomy [26], the cases described above concern training data quality; in general, SAFE can help engineers to discover any fault that affects the correctness of the DNN output (e.g., a missing model layer). However, we do not integrate mechanisms to automatically determine the underlying cause for the observed failures. In general, we suggest engineers to first inspect root cause clusters to determine major pitfalls (e.g., missing output class) then proceed with automated retraining (i.e., Steps 3-6); if the DNN accuracy is not sufficiently improved then it is necessary to modify the DNN (e.g., add layers or change architecture).

### 3.6 Step 4: Identify unsafe images

Since the manual labeling of images is expensive, it is necessary to automatically identify an unsafe set of reduced size containing images that can improve the DNN accuracy to achieve a

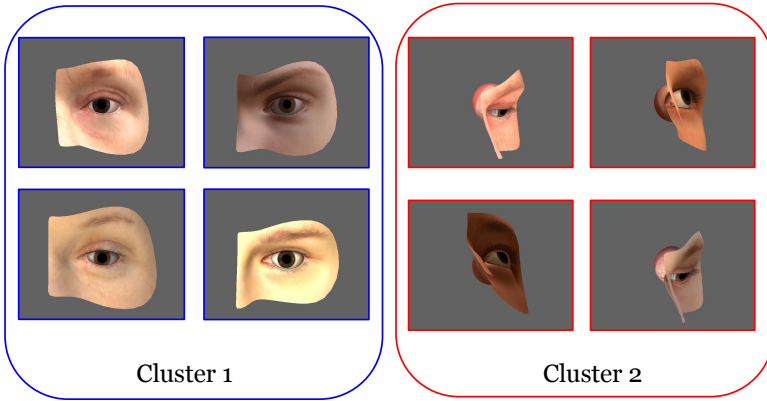


Fig. 7. Examples of root cause clusters for Gaze detection

cost-effective retraining process. An unsafe set is a subset of unlabelled images from the improvement set, to be labeled and used for retraining the model. Unlike HUDD, SAFE relies on a new clustering-based selection method to automatically select this unsafe set. We identify images from the improvement set that belong to a root cause cluster and select images that, within the same cluster, are representative of the different types of images in the cluster (e.g., different face shapes with a gaze angle that confuses a gaze classifier). Therefore, we assume that the selected images are representative of a cluster in the sense that they contain sufficient information to replace the other cluster members [45] and can be effectively used for retraining.

The key difference between SAFE and HUDD, in this step, is that the latter relies on a Hierarchical Clustering Algorithm. This method only finds spherical clusters. In addition, HUDD depends on the cluster medoids and the cluster ratio to determine if improvement set images belong to the root cause clusters. Unfortunately, such a method might be suboptimal if clusters are not spherical. With DBSCAN, the identification of core points enables the determination of representative cluster members even in non-spherical clusters. It does not rely on cluster centroids but core points in each cluster, assuming that, taken together, these points represent the entire cluster. A point close to any core point will be assigned to the cluster represented by this core point.

In SAFE, the selection is performed according to Algorithm 1, which is detailed below. After selecting the unsafe set, we follow the same retraining process as HUDD. We retrain the DNN model by relying on a training set consisting of the union of the original training set and the manually labeled unsafe set. The original training set is reused to prevent reducing the accuracy of the DNN for parts of the input space that are safe. The retraining process is thus expected to lead to an improved DNN model.

To minimize the retraining effort, and most particularly that related to labeling, we look for representative images in each cluster. For that, we rely on the core points (see Section 3.4). The choice of the core points is motivated by the fact that they represent better the shape of a particular cluster than a centroid. As described in Section 3.4, a root cause cluster is typically represented by several core points. The border points that define the cluster's shape are localized around the core points. We assume that the points close to the core points contain enough information to replace the other border points. These points usually take approximately the same shape as the cluster [34]. Recall that core points, unlike centroids, can represent a non-convex cluster with arbitrary shapes.

**Algorithm 1** SAFE Unsafe Set Selection Algorithm**Input:** improvement set images  $\mathcal{X}$ , core points detected for each cluster, a set of clusters  $\mathcal{C}$ .**Output:** unsafe set

- 
- 1: **for**  $x$  in  $\mathcal{X}$  **do**
  - 2:     Assign  $x$  to the cluster corresponding to the closest core point.
  - 3: Compute  $N$ , the number of selected images for the unsafe set based on Equation 3.
  - 4: Compute the number of selected images  $r_i$  from each cluster  $c_i$ , computed as  $N$  over the proportion of images in the cluster.
  - 5: **for**  $c_i$  in  $\mathcal{K}$  **do**
  - 6:     Sort the images in ascending distance to their respective closest core point.
  - 7:     From the sorted images, take the  $r_i$  first images and add them to the unsafe set.
- 

Algorithm 1 show the steps for the selection of the unsafe set for retraining. The algorithm requires the root cause clusters and their core points. It also requires an improvement set.

The algorithm starts by assigning every image in the improvement set to the closest core point in terms of euclidean distance (line 1, Algorithm 1).

On lines 2 and 3, SAFE computes  $N$ , the number of images to be selected for the unsafe set and the number of images to be selected from each root cause cluster  $i$ , denoted as  $r_i$ , where  $r_i = N \times \frac{C_i}{C}$ .  $C$  is the number of images in the improvement set and  $C_i$  is the number of such images assigned to cluster  $i$ . Unsafe set images across root cause clusters therefore preserve the distribution of the improvement set across such clusters.

As in HUDD, we assume that the distribution of error-inducing images across clusters is similar in the improvement and test sets. We determine the number of images  $N$  selected from the improvement set to include in the unsafe set as follows:

$$N = (|\text{TestSet}| \times sf) \times (1 - \text{TestSetAcc}) \quad (3)$$

$|\text{TestSet}|$  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, same as HUDD to make the comparison fair).  $\text{TestSetAcc}$  represent the accuracy of the original model on the test set of the case study subject. The term  $(1 - \text{TestSetAcc})$  indicates the proportion of error-inducing images that are observed in the improvement set.  $(|\text{TestSet}| \times sf) \times (1 - \text{TestSetAcc})$  estimates the number of error-inducing images that should be selected from the improvement set. The term  $(|\text{TestSet}| \times sf)$  provides an upper bound for the unsafe set size as a proportion of the test set size.

In line 4, the images are sorted according to ascending distance to their respective closest core point. Finally, in line 5, for each cluster  $i$ , we select the first  $r_i$  sorted images to include in the unsafe set.

We also illustrate the algorithm steps in Figure 8. The first image represents the core points obtained from the clustering of the error-inducing images (the dots represent the core points). In the second step, we assign the images in the improvement set to their closest core point, respectively. Then, we select a subset of points from the neighborhood of each core point based on Algorithm 1. The last image represents the selected unsafe set.

Our algorithm excludes from the unsafe set the images leading to DNN errors due to root causes not observed in the test set; indeed, such images will be distant from clusters' core points. Furthermore, such images will not help improve the DNN performance on the test set, which is our objective since the test set is assumed to be representative of real-world scenarios. Finally, when the improvement set does not include any image belonging to a root cause cluster then SAFE does

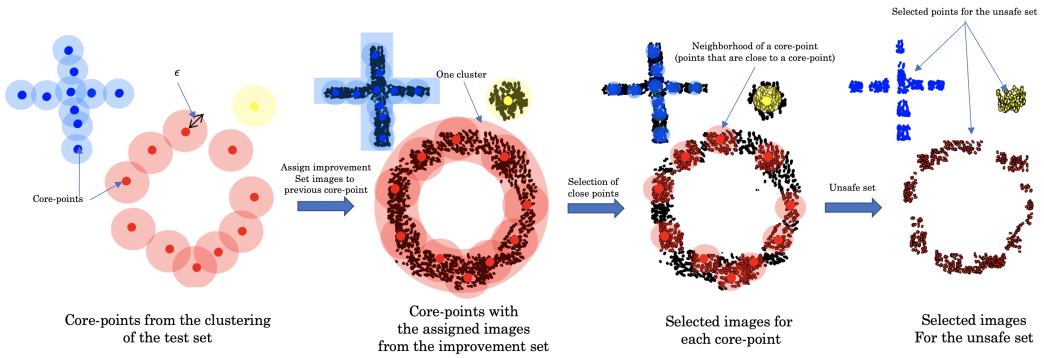


Fig. 8. Unsafe set selection

not assign any image to the cluster; this is not the case for HUDD which, different from SAFE, will not prevent engineers from labeling useless images.

### 3.7 SAFE running example

This section presents an example of SAFE usage. It is based on the headpose detection DNN (HPD) considered in our empirical evaluation. HPD receives as input a picture taken from a camera positioned inside a car; the picture is automatically cropped to a size of  $640 \times 640$  pixels. HPD classifies the head position according to nine classes: straight, turned bottom-left, turned left, turned top-left, turned bottom-right, turned right, turned top-right, reclined, looking up. Figure 9 provides an example of an image classified by SAFE.

To reduce development costs, HPD has been trained and tested using images generated by a simulator capable of generating pictures of human heads. The training and test sets consists of 16013 and 2825 images, respectively, both generated by randomly selecting simulator parameters' values. The training and the test set could also have included real-world images. After 30 epochs, we obtained an accuracy of 88.03%. From the test set, 1580 images were misclassified, they represent the error-inducing images that should be investigated to determine the root causes of errors.

We implemented the SAFE pipeline as a Jupyter Notebook<sup>3</sup>.

The SAFE pipeline starts by pre-processing the error-inducing images to match our model's input requirement (SAFE Step 1.1, in Figure 4). It automatically converts the images into a NumPy<sup>4</sup> array and downsample them as explained in Section 3.1.

After preprocessing, the images are ready for feature extraction (SAFE Step 1.2, in Figure 4). The SAFE pipeline automatically extracts the features by relying on a pre-trained VGG model loaded by the Notebook. Precisely, for each image, SAFE stores the output of the second-last fully connected layer of the VGG model, which leads to an array of 512 features for each image.

The pipeline continues by applying the PCA dimensionality reduction method to reduce the number of features from 512 to 256 (SAFE Step 1.3, in Figure 4). The output of the PCA method is an  $1580 \times 256$  array, where 1580 is the number of error-inducing images and 256 is the number of features. The number of features (i.e., 256) has been empirically determined in preliminary experiments (see Section 3.3) and is not supposed to be further modified by end-users. This array is passed to DBSCAN as an input. We use DBSCAN from the SciKitLearn library<sup>5</sup>.

<sup>3</sup><http://jupyter-notebook.readthedocs.io>

<sup>4</sup><http://numpy.org>

<sup>5</sup><http://scikit-learn.org>



Fig. 9. Example images from the HPD dataset

Before performing clustering (SAFE Step 1.4), we first need to choose the optimal parameter settings. We apply the method explained in Section 4.2 to obtain  $\epsilon = 0.9$  and  $minPts = 9$ . Using these parameters, we run our algorithm to generate the final root cause clusters, which are 20 in this case. The next step of the pipeline (i.e., SAFE Step 1.5) includes a procedure that, from the clusters generated by DBSCAN, generates several folders, each containing the images belonging to the cluster. It also generates an animated gif image (similar to a video) with the images belonging to each cluster, to help the user visualize it. The end-user is then expected to visualize a portion of the images appearing in each cluster (e.g., five according to our experimental results in Section 4.2.5) to perform the functional safety analysis step of SAFE (i.e., Step 2 in Figure 3). Example images are reported in Figure 6.

Next, the end-user should provide an improvement set (SAFE Step 3). For our experiments with HPD, we rely on an improvement set of 4103 images generated with additional executions of the simulator. We could have also included real-world images collected by our industry partner in the field but they might have prevented replicability (e.g., they cannot be publicly shared because of privacy agreements). In our execution, SAFE selected 154 images as an unsafe set for retraining. The number of selected images is computed automatically based on the selection factor ( $sf$ ) configured by the end-user (see Section 3.6) These images need to be labelled by the end-user (SAFE Step 5) but for our case study subject, labels are automatically derived from simulator parameters. In case the improvement set includes real-world images, labelling is performed manually.

After obtaining the unsafe set, the end-user simply combines it with the training set and retrains the model according to the specific procedure of the DNN under analysis.



## 4 EMPIRICAL EVALUATION

In this section, we aim to evaluate our approach. SAFE is expected to perform better than HUDD in terms of the quality of the clustering and DNN accuracy after retraining. We choose HUDD for comparison not only because SAFE aims to improve over it but because they are the only approaches in the literature that aim to support safety analysis, which is achieved through the identification of root cause clusters and the selection of images for retraining based on these clusters. To investigate whether if such expectations hold and SAFE is useful, we compare these two approaches following the experimental procedures described below addressing the following research questions.

**RQ1.** Does SAFE enable engineers to identify the root causes of DNN errors? The clusters produced by SAFE should provide useful information to identify plausible causes of DNN errors in a form that is amenable to practical analysis.

**RQ2** Does SAFE enable engineers to more effectively and efficiently retrain a DNN when compared with HUDD and baselines approaches? We expect SAFE to lead to a higher model accuracy after retraining.

**RQ3** Does SAFE provides time and memory savings compared to HUDD? SAFE's black-box nature should provide significant time and memory savings, compared to HUDD, which is a white-box approach.

To perform our empirical evaluation, we have implemented SAFE as a toolset that relies on the PyTorch [54] and SciPy [60] libraries. Our toolset, case study subjects, and results are available for download [5]. In our experiments, steps 1 to 5 were carried out on an Intel Core i9 processor running macOS with 32 GB RAM. Step 6 (retraining) was conducted on the HPC facilities of the University of Luxembourg (see <http://hpc.uni.lu>). We relied on a Dual Intel Xeon Skylake CPU (28 cores) and 128 GB of RAM.

### 4.1 Subjects of the study

We rely on images generated using simulators as it allows us to associate each generated image to values of the simulator's configuration parameters. 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. Such simulators are increasingly common, and of higher fidelity in many domains [49], including automotive and aerospace.

We consider the same DNNs as the HUDD paper, which support gaze detection, drowsiness detection, headpose detection, and face landmarks detection systems under development at IEE Sensing, our industry partner.

Eye gaze detection systems (GD) use DNNs to perform eye tracking. Gaze tracking is typically employed to determine a person's focus and attention. It classifies the gaze direction into eight classes (i.e., TopLeft, TopCenter, TopRight, MiddleLeft, MiddleRight, BottomLeft, BottomCenter, and BottomRight). The drowsiness detection system (OC) features the same architecture as the gaze detection system, except that the DNN predicts whether eyes are opened or closed.

The headpose detection system (HPD) is an important cue for scene interpretation and remote computer control like driver assistance systems. It determines a head pose in an image 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) 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. Several face landmarks match each face element.

Table 1. Case Study Systems

DNN	Data Source	Training Set Size	Test Set Size	DNN Accuracy	number of error inducing images
GD	UnityEyes	61,063	132,630	95.95%	5371
OC	UnityEyes	1,704	4,232	88.03%	506
HPD	Blender	16,013	2,825	44.07%	1580
FLD	Blender	16,013	2,825	44.99%	1554
OD	CelebA [42]	7916	5276	84.12%	838
TS	TrafficSigns [28]	29,416	12,631	81.65%	2317

GD, OC, and HPD follow the AlexNet architecture [36] which is commonly used for image classification. FLD, which addresses a regression problem, relies on an Hourglass-like architecture [50].

Since SAFE can be applied to DNNs trained using either a simulator or real images, we also considered additional DNNs trained using real-world images. To do so, we selected the same DNNs included in the HUDD paper, which target traffic sign recognition (TS) and object detection (OD), and are typical features in automotive, DNN-based systems.

TS recognizes traffic signs in pictures whereas OD determines if a person wears eyeglasses. The latter has been selected to compare results with MODE, a state-of-the-art retraining approach whose implementation is not available, but with an objective that is close to that of SAFE. Both TS and OD follow the AlexNet architecture [36].

We further describe in Table 1 the case study subjects used to evaluate SAFE. We indicate either the simulator used to generate the data or the real-world image dataset. The data is then randomly split into training and test sets whose sizes are reported. Further, we report the number of error-inducing images, which are the images from the test set leading to a result different than the ground truth. For classifier DNNs (GD, OC, HPD, OD, TS), DNN errors are inconsistent predicted and expected classes. For FLD, we determine a DNN error when the average distance of the predicted keypoints is above four pixels, as suggested by IEE engineers.

Since DNN errors and, consequently, clustering results, depend on the initial training of the DNN under analysis, to deal with such randomness we repeated the initial training ten times for the case study subjects GD, OC, and HPD; each training execution relies on a different split of the training and the test data sets. Unfortunately, we could not repeat the training for the FLD DNN because it was provided by our industry partner along with the error-inducing images. Further, we could not repeat the execution of HUDD ten times for each case study DNN because of the large amount of time required to compute distance matrices based on heatmaps (see Section 4.2.7). However, to discuss the statistical significance of the differences between HUDD and SAFE, for each of the metrics selected to address our research questions, we relied on a one-sample Wilcoxon signed rank test. The one-sample Wilcoxon signed rank test is a non-parametric statistical hypothesis test used to determine whether the median of a population (here, the SAFE median) is greater than a reference value (here, the HUDD result). It enables us to test the null hypothesis: *the SAFE median is equal to the HUDD result*.

Finally, to determine the number of components to be selected by PCA, which is an input parameter for SAFE, we conducted a set of experiments. Precisely, we considered a number of features between 2 and 256, considering all powers of 2; then we applied DBSCAN and measured the quality of its result based on the Silhouette Index [57]. We performed the analysis on all the case studies and concluded that 256 is the number of features that provides the best results.

Table 2. Root cause clusters generated by SAFE.

	SAFE					HUDD		
	# error inducing images	# of clusters		% of inspected images		# error inducing images	# of clusters	% of inspected images
		min/max/med	min/max/med	p-value	min/max/med			
GD	4967/6290/5602	14/31/25	0.004	1.41/2.46/2.24	0.004	5371	16	1.49
OC	409/557/492	21/33/26	0.002	25.67/29.62/26.15	0.002	506	14	13.83
HPD	1371/2089/1519	20/30/24	0.002	7.29/7.18/7.99	0.002	1580	17	5.38
FLD	1554	64	/	20.5	/	1554	71	22.84
OD	758/933/822	2/2/2	0.002	1.07/1.32/1.22	0.004	838	14	8.35
TS	2239/2698/2450	7/12/9	0.004	1.45/2.63/1.93	0.004	2317	20	4.31

## 4.2 Experimental Results

We refine RQ1 into four complementary subquestions (RQ1.1, RQ1.2, RQ1.3, RQ1.4, and RQ1.5), which are described in the following, along with their corresponding experiment design and results.

### 4.2.1 RQ1.1. Is the number of generated clusters small enough for enabling visual inspection?

*Design and measurements.* Though this is to some extent subjective and context-dependent, we discuss whether SAFE finds a number of root cause clusters that is amenable to inspection by experts. To respond to this research question, we assume that experts visually inspect five images per root cause cluster to be able to make a decision. This assumption is supported by an experiment we conducted, as presented in Section 4.2.5. Under this assumption, we compare SAFE and HUDD in terms of the number of generated clusters and based on the ratio of error-inducing images that should be visually inspected when relying on each method. This ratio is calculated as follows:

$$ratio = \frac{(k \times 5) \times 100}{n} \quad (4)$$

where  $k$  is the number of root cause clusters, and  $n$  is the number of error-inducing images.

*Results.* Table 2 shows, for each case study subject, the total number of error-inducing images belonging to the test set, the number of root cause clusters generated by SAFE and HUDD, and the ratio of error-inducing images that should be visually inspected when using SAFE or HUDD.

For the respective DNNs, SAFE identifies 25 (GD), 26 (OC), 24 (HPD), 64 (FLD), 2 (OD), 9 (TS) root cause clusters (for GD, OC, HPD, OD, and TS we refer to median of the ten runs executed). In contrast, HUDD identifies 16 (GD), 14 (OC), 17 (HPD), 71 (FLD), 14 (OD), and 20 (TS) root cause clusters.

We notice that in 50% of the case study subjects (see bold values in Table 2), SAFE yields a lower number of clusters. This experiment shows both SAFE and HUDD find an acceptable number of clusters for visual inspection. Indeed, the ratios of error-inducing images to be inspected are low (between 1.07 and 26.15, with a median of 5.12). These results suggest that using SAFE can save significant effort compared to the manual inspection of the entire set of images.

### 4.2.2 RQ1.2. Does SAFE generate root cause clusters with a significant reduction in variance for simulator parameters?

*Design and measurements.* This research question investigates if SAFE achieves high within-cluster similarity concerning at least one simulator parameter. Indeed, since we rely on DNNs that are trained and tested with simulators, images assigned to the same cluster should present similar values for a subset of the simulator parameters. Within each cluster, the variance of these

Table 3. Image parameters considered to address RQ1.2

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.

parameters should be significantly smaller than that computed on the entire test set. For a cluster  $C_i$ , the rate of reduction in variance for a parameter  $p$  can be computed as follows:

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

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

Table 3 provides the list of parameters considered in our evaluation.

In the case of GD and OC, we rely on the parameters given by the simulator, except for the ones that capture coordinates of single points used to draw pictures (e.g., eye landmarks) since these coordinates alone are not informative about the elements in the image. We also rely on parameters that are derived from the coordinates mentioned above and capture characteristics that are potentially related to error-inducing images: PupilToBottom, PupilToTop, DistToCenter, Openness.

For HPD, we also considered the parameters provided by the simulator, omitting the landmark coordinates. Parameters expressed with X-Y-Z coordinates are considered as three separate parameters (e.g., Headpose).

As for FLD, since a DNN error may depend on the specific shape and expression of the processed face (i.e., the particular 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).

Note that simulator parameters are only used to objectively evaluate the approach; they are not involved in the practical application of the approach. Section 4.2.6 addresses the application of SAFE to case study subjects for which a simulator is not available (i.e., TS and OD).

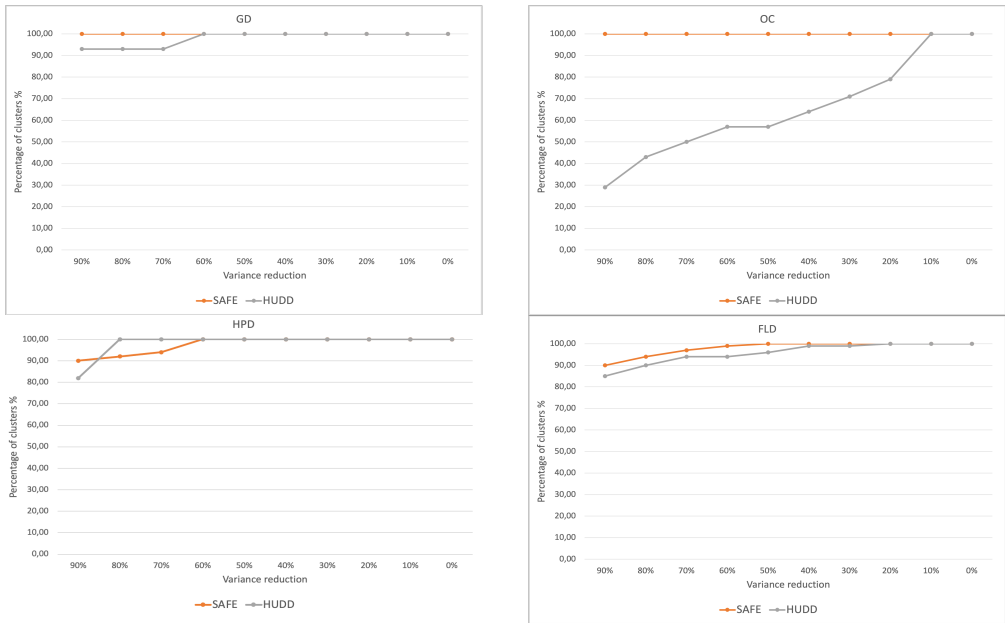


Fig. 10. RQ1.2: median percentage of clusters with at least one parameter showing a reduction rate above a threshold in the range [0% - 90%].

Since the number of parameters that capture common error causes is not known a priori, we consider a significant variance reduction in at least one parameter to be enough for the cluster to be indicative of root causes. Therefore, we compute the percentage of clusters showing such a variance reduction for at least one of the parameters.

Consistent with HUDD, we compute the percentage of clusters with a variance reduction between 0% and 90%, with incremental steps of 10%. To answer our research question positively, a high percentage of the clusters should reduce variance for at least one of the parameters. We compare our results to those of HUDD.

*Results.* We report in Table 4 the maximum, minimum, and median percentage of clusters having at least one parameter showing a reduction rate above a threshold in the range [0% - 90%], compared to the percentage obtained by HUDD. We also report the p-values resulting from performing a one-sample Wilcoxon signed rank test (see Section 4.1). We notice that, when the median obtained with SAFE is higher than the value obtained with HUDD (i.e., SAFE performs better), the p-values are always below 0.05. This implies that, in these cases, the median percentage of clusters with a reduced variance obtained with SAFE is significantly larger than the one obtained with HUDD.

To enable visual comparison, Figure 10 plots the median percentage of clusters with variance reduction for at least one of the simulator parameters, with different reduction rates, for both SAFE and HUDD. Results show that SAFE yields a higher percentage for three out of the four case study subjects (i.e., GD, OC, FLD). For HPD, based on the p-values reported in Table 4, the differences between HUDD and SAFE are not significant (i.e., HUDD does not perform better than SAFE).

Table 4. Minimum, Maximum and Median percentage of clusters with at least one parameter showing a reduction rate above a threshold in the range [10% - 90%], with the percentage obtained by HUDD and the p-value when comparing SAFE to HUDD.

Threshold		GD	OC	HPD	FLD
90%	SAFE min/max/median	100%/100%/100%	100%/100%/100%	85%/100%/90%	90%
	HUDD	93%	29%	82%	85%
	p-value	0.001	0.001	0.004	
80%	SAFE min/max/median	100%/100%/100%	100%/100%/100%	85%/100%/92%	94%
	HUDD	93%	43%	100%	90%
	p-value	0.001	0.001	0.99	
70%	SAFE min/max/median	100%/100%/100%	100%/100%/100%	87%/100%/94%	97%
	HUDD	93%	50%	100%	94%
	p-value	0.001	0.001	0.99	
60%	SAFE min/max/median	100%/100%/100%	100%/100%/100%	100%/100%/100%	99%
	HUDD	100%	57%	100%	94%
	p-value	1	0.001	1	
50%	SAFE min/max/median	100%/100%/100%	100%/100%/100%	100%/100%/100%	100%
	HUDD	100%	57%	100%	96%
	p-value	1	0.001	1	
40%	SAFE min/max/median	100%/100%/100%	100%/100%/100%	100%/100%/100%	100%
	HUDD	100%	64%	100%	99%
	p-value	1	0.001	1	
30%	SAFE min/max/median	100%/100%/100%	100%/100%/100%	100%/100%/100%	100%
	HUDD	100%	71%	100%	99%
	p-value	1	0.001	1	
20%	SAFE min/max/median	100%/100%/100%	100%/100%/100%	100%/100%/100%	100%
	HUDD	100%	79%	100%	100%
	p-value	1	0.001	1	
10%	SAFE min/max/median	100%/100%/100%	100%/100%/100%	100%/100%/100%	100%
	HUDD	100%	100%	100%	100%
	p-value	1	1	1	

Table 5. Number of core-points and number of clusters for each case study subject

Case Study Subject	# of core points (min/max/median)	# of clusters (min/max/median)	% of images considered core points (min/max/median)
GD	2389/5808/5080	14/31/26	46%/92%/88%
OC	366/495/441	21/33/26	89%/97%/90%
HPD	259/623/316	20/30/24	18%/30%/22%
FLD	622	64	40%

We report in Table 5 the minimum, maximum and median number of core points found by the clustering algorithm for each case study subject and the ratio of error-inducing images being considered as core points. We recall that a cluster consists of several core points and that the core points determine the cluster shape. The ratio of error-inducing images being considered as core points is therefore an indicator of the complexity of the cluster shape. The larger this ratio, the more complex the cluster shape and the less likely it is to be convex. Since non-convex clusters cannot be properly modeled by a centroid-based algorithm or even a hierarchical clustering algorithm (see Section 3), this partly explains the results presented in Figure 10.

For GD and FLD, SAFE performs slightly better than HUDD. With GD, out of ten executions, SAFE yields a median of 100% of the clusters with a variance reduction above 90% compared to 93%

for HUDD. As for FLD, it obtained 90% of the clusters with variance reduction above 90%, compared to 85% for HUDD. These values are close because the number of clusters detected by both methods is similar for GD and FLD. Detecting the optimal number of clusters is crucial as it leads to root cause clusters grouping very similar images with less noise. Consequently, identifying the right root cause clusters will result in higher variance reduction. Nevertheless, the slight superiority shown by SAFE is explained by the fact that the latter finds root cause clusters with arbitrary shapes, compared to convex shapes found by HUDD. This is important since arbitrary-shaped clusters can find more homogeneous clusters (i.e., clusters with higher within-cluster similarity) with very similar images. In contrast, a convex cluster tends to be less dense and can group rather dissimilar images. In the case of OC, the percentage of clusters with a given variance reduction obtained by SAFE is much higher than that obtained by HUDD. SAFE yielded 100% of the clusters (median of ten executions) with variance reduction above 90%, in contrast to 29% for HUDD. This is explained by the fact that SAFE found a much higher number of clusters than HUDD (26 for SAFE compared to 14 for HUDD). Also, 90% of the error-inducing images are considered core points by SAFE (median), thus indicating complex cluster shapes, which may, in turn, explain the detection of more clusters. A larger number of clusters leads to root cause clusters with a lower number of images (15 images per cluster on average for SAFE with OC), which have higher chances to contain similar images.

For the HPD case study subject, HUDD and SAFE results are very close. SAFE yields 90% of the clusters (median of ten runs) with variance reduction above 90% compared to 82% for HUDD. At the same time, HUDD shows 100% of the clusters with variance reduction above 80%. Both methods yield 100% of the clusters with variance reduction above 60%. We notice that the number of clusters detected by both methods is pretty close (24 for SAFE compared to 17 for HUDD), thus explaining these results. Also, we observe that only 22% of the error-inducing images are considered core points, hence indicating that clusters do not have complex shapes and are closer to being convex than in the OC case, for example.

In general, when the cluster shapes obtained by SAFE are relatively simple, the results of the two approaches can be expected to be similar. In contrast, when clusters have arbitrarily complex shapes, there is a clear advantage in using SAFE, as illustrated by the OC results and to a lesser extent the GD results.

In general, in spite of being a black-box approach, SAFE tends to find a high percentage of clusters with at least one reduced parameter. For GD and OC, 100% of the clusters show parameters with a variance reduction above 90%. HPD and FLD yield both 90%. Based on these results, we can positively answer RQ1.2 since all clusters present at least one parameter with a positive, significant reduction rate ( $> 50\%$  in Figure 10).

#### 4.2.3 RQ1.3. Do parameters with high variance reduction represent a plausible cause for DNN errors?

*Design and measurements.* This research question investigates whether SAFE helps engineers understand the root causes for each error.

We assume that DNN errors occur in specific areas of the simulator parameter space. Under this assumption, we identify a set of unsafe parameters and corresponding unsafe values around which a DNN error is susceptible to occur. Table 6 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 distinguishing labels pertaining to different gaze directions. These unsafe parameters were selected in a systematic manner for all the case study subjects. Precisely, we report as unsafe values all the values used to label different classes (e.g., eyes openness above/below 20 pixels) and values for borderline cases (i.e., cases in which portions of the face are hidden). Also,

Table 6. 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).
HPD	PupilToTop	Value below which the pupil is mostly above the eyelid (i.e., -16 pixels).
	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°).

we have identified additional unsafe parameters that can cause a DNN error because they lead to masked elements in images (e.g., distance between the pupil center and the iris center, distance between the pupil bottom and the bottom eyelid margin). Note that determining unsafe parameters and values is only required for experimental purposes here, as detailed below, and not for applying SAFE in practice.

Face images generated with such unsafe values may lead to DNN errors because the DNN either cannot distinguish two classes or because part of the human face is not present in the image. Therefore, we expect all the error-inducing images having such characteristics to belong to an appropriate cluster; precisely, they should belong to a cluster having (a) high variance reduction for the unsafe parameter and (b) an average value close to the identified unsafe value.

For our experiment, we consider that a root cause cluster is explanatory in terms of root causes if it satisfies two requirements: (1) It should have at least one unsafe parameter with a variance reduction above 50%, (2) the cluster average should be close to one unsafe value. For Gaze Angle, Openness, Headpose-X, and Headpose-Y, an average value is considered close to an unsafe value if the difference between them is below 25% of the length of the subrange including the average value. For DistToCenter, PupilToBottom, and PupilToTop, an average value is considered close to an unsafe value if it is below or equal to it. For the FLD case study subject, since the reason for not detecting a landmark cannot be related to a single simulator parameter but often depends on combinations of parameters (e.g., the position of the head and the illumination angle lead to shadows on the face), it is impossible to determine unsafe values and therefore such a set of explanatory parameters; as a result, FLD is omitted from this experiment.

Based on the above, we address this research question by computing the percentage of clusters that are explanatory according to our definition. The higher this percentage, the more evidence we have that clustering is useful for identifying causes of DNN errors.

*Results.* Table 7 shows the percentage of the root cause clusters that are explanatory for both SAFE and HUDD. Since we repeat the execution SAFE with ten different DNN instances for each case study subject, we report the minimum, maximum and median of the percentages obtained. Across all three case study subjects, SAFE shows a higher percentage of explanatory root cause clusters than HUDD. The median results with GD, OC, and HPD are 86%, 100%, and 90%, respectively, compared to 86%, 57%, and 88%, respectively, with HUDD. Table 7 also reports the p-values resulting from performing a one-sample Wilcoxon signed rank test (see Section 4.1). The p-values are below 0.05 for OC and HPD, which indicates that the percentage of SAFE's root cause clusters that present



at least one explanatory parameter is significantly larger than the one obtained with HUDD for OC and HPD. As for GD, the results are similar.

Table 7. Minimum, Maximum and Median percentage of root cause clusters that present at least one explanatory parameter with the percentage obtained by HUDD and the p-value when comparing SAFE to HUDD.

Case Study Subjects	SAFE			HUDD	p-value
	Min	Max	Median		
GD	83%	100%	86%	86%	0.99
OC	93%	100%	100%	57%	0.002
HPD	84%	96%	90%	88%	0.02

These results show a large difference in the percentage of clusters that can be explained between SAFE or HUDD for the OC case study subject (43% difference). Indeed, for SAFE, all the clusters have a high reduction in variance, while this is only the case for 57% of the clusters for HUDD. As explained in the previous Section, this can be explained by the fact that OC clusters have a complex shape, more so than in other case study subjects.

As for GD and HPD, the SAFE median is close to the result obtained with HUDD (although for HPD the difference is significant with a significance level of 0.05). For the median, we observe a 2% difference for HPD and no difference for GD. These results confirm the results obtained in RQ1.2. For GD and HPD, both methods show 100% of the clusters with a parameter presenting a variance reduction above 50%. These results are however still slightly in favor of SAFE. Once again, the above results are explained by the fact that the root cause clusters found by SAFE can take arbitrary shapes. Such clusters, as previously explained and in the general case, have better chances to group similar images than clusters with convex shapes.

#### 4.2.4 RQ1.4. Does SAFE identify more distinct error root causes than HUDD?

*Design and measurements.* This research question investigates if SAFE identifies a larger number of possible causes of errors than HUDD. Specifically, we compare the two approaches in terms of the number of unsafe values being covered by at least one cluster. We say that an unsafe value  $v$  is covered by a cluster  $c$ , when  $c$  presents a parameter  $p$  with a high variance reduction and the parameter  $p$  has an average value close to the unsafe value  $v$ .

Since our simulators generate images having parameter values that are uniformly sampled within the input domain, every unsafe value has the same likelihood of being observed in the test set images. Therefore, ideally, we aim for the root cause clusters to cover all such values.

*Results.* In Table 8, we report the minimum, maximum and median percentage of the unsafe values covered by the root cause clusters obtained when applying SAFE to our case study subjects. The clusters generated by SAFE with GD, OC, and HPD cover (median) 92%, 64%, and 80% of the unsafe values, respectively. The clusters generated by HUDD, instead, cover 71%, 50%, and 60% of the unsafe values, respectively. The p-values resulting from performing the one-tailed, one-sample Wilcoxon signed-rank test are always below 0.05, which implies that the median obtained with SAFE is significantly higher than the result obtained with HUDD.

Table 8. Minimum, Maximum and Median percentage of the unsafe values covered by the root cause clusters with the p-value when comparing SAFE to HUDD.

	SAFE			HUDD	p-value
	Min	Max	Median		
GD	85%	100%	92%	71%	0.002
OC	64%	71%	64%	50%	0.002
HPD	60%	80%	80%	60%	0.004

Table 9. Coverage of the unsafe values by the root cause clusters (OC and GD case study subjects)

	Unsafe values	GD		OC	
		SAFE	HUDD	SAFE	HUDD
Angle:	337,5	✓	✓	✓	✗
	22,5	✓	✓	✗	✗
	67,5	✓	✓	✗	✗
	112,5	✓	✓	✗	✗
	157,5	✓	✓	✗	✗
	202,5	✓	✓	✓	✗
	247,5	✓	✓	✓	✓
	292,5	✓	✓	✓	✓
H-Headpose	220	✓	✗	✓	✓
	160	✓	✓	✓	✓
V-Headpose	20	✓	✗	✓	✓
	340	✓	✗	✓	✗
StrangeDist Top/Bot	-14	✓	✗	✓	✓
Distance	25	✓	✓	✓	✓
TOTAL Coverage		<b>14</b>	10	<b>10</b>	7

Table 10. Coverage of the unsafe values by the root cause clusters (HPD case study subject)

		HPD	
		SAFE	HUDD
H-Headpose	-28	✓	✓
	-10	✓	✓
	0	✓	✓
	10	✓	✗
	21	✗	✗
V-HeadPose	-88	✗	✗
	-10	✓	✓
	0	✓	✓
	10	✓	✓
	77	✓	✗
TOTAL Coverage		<b>8</b>	6

Below, we discuss, more in detail, the differences between SAFE and HUDD. To exemplify our discussion, we report in Table 9 (GD, OC) and Table 10 (HPD) examples of unsafe values covered by the clusters generated in one of the runs of SAFE and HUDD.

Based on Table 8, for GD, SAFE identifies root cause clusters covering 92% unsafe values (median out of ten runs), compared to 71% for HUDD. This is because SAFE relies on images represented with features extracted from convolutional layers, which provide a better representation than heatmaps and show aspects that are not captured by heatmaps, such as eye shape, edges, and corners.

For OC, SAFE covers a median of 65% unsafe values compared to 50% for HUDD. The uncovered unsafe values concern the parameter *Angle*. However, in the OC case study subject, we mainly focus on eye openness (*Distance* parameter) and the distance between the pupil and eyelid (*StrangeDist Top/Bot* parameter). All of the unsafe values for these two relevant parameters were covered by SAFE and HUDD.

For the HPD case study subject, SAFE covers a median of 80% unsafe values compared to 60% for HUDD. None of the techniques covers values 21 for *H-Headpose* and  $-88$  for *V-Headpose*. However, such values can be observed if we also consider parameters with a variance below 50% (which is an arbitrary threshold). These two values represent boundary values that we believe confuse SAFE as they correspond to situations where it is hard to see the eyes and the shape of the head. As a result, such images are sometimes clustered erroneously. Thus, these clusters show a lower variance reduction.


In addition, we report that 90% of the clusters cover a unique set of unsafe values (e.g., *Angle*=337.5, *H-Headpose*=220, *V-Headpose*=340, *Distance*=25). Concerning the remaining 10%, we observe that they are still unique but differ with respect to a parameter that is not unsafe. Therefore, we conclude that all the clusters generated by SAFE cover a unique set of parameter values that are useful to determine distinct failure causes.

#### 4.2.5 RQ1.5. How many images are required to identify commonalities in a cluster?

*Design and measurements.* We aim to determine how many images from a root cause cluster an engineer should inspect to correctly identify the root cause captured by the cluster. Recall that in our previous analysis (i.e., RQ1.1), we made an assumption regarding this point to estimate cost savings that can be expected from SAFE. To this end we conducted an online questionnaire-based experiment, following best practices [41]. Our population of participants consists of 19 PhD students at the University of Luxembourg and University of Ottawa. All the selected students hold a master's degree in computer science or a corresponding engineering degree; further, most have acquired fundamentals in machine learning and a few work on safety-related topics. Therefore, all the selected participants are competent to perform the experimental task, which does not require background knowledge, though they are less familiar with the problem domain than engineers would be in practice and therefore can be expected to make more mistakes.

In our experiment, we asked participants to identify the commonalities in a randomly selected subset of images belonging to randomly selected root cause clusters generated by SAFE for our case study subjects. Such task emulates what should be done by an engineer to understand the root causes for an error. We asked each participant to perform the task three times, on different clusters of images. Each cluster of images included respectively 5, 10, and 15 images randomly selected from a root cause cluster belonging to a different case study subject. Also, for the same subject, each participant received a different random set of images, belonging to a randomly selected cluster. In short, each participant therefore received three questions with 5 images, 10 images, and 15 images, respectively, belonging to different clusters from different case study subjects.

Choose the commonalities for this set of images.



[Check all that apply](#)

- The eyes are abnormally open
- The eyes are looking middle centre
- The eyes are looking between the top left and the middle left of the image
- The eyes are looking between the top centre and the top left of the image
- The eyes are looking between top right and top centre
- The eyes are looking between the middle right and the top right of the image
- The eyes are looking between the bottom right and the middle right of the image
- The eyes are looking between the bottom centre and the bottom right of the image
- The eyes are looking between bottom left and bottom centre
- The eyes are looking between the middle left and the bottom left of the image
- The pupil is mostly above the eyelid
- The pupil is mostly under the eyelid
- I cannot identify any commonality

Fig. 11. Example question appearing in our questionnaire.

We provided closed-ended questions to objectively compare the obtained answers with the ground truth (i.e., likely root causes for DNN errors). Similarly to RQ1.4, we considered only case study subjects performing classification tasks because, for these cases, the availability of the ground truth makes it possible to determine the reasons for DNN failure objectively.

As commonalities to be identified by the participants, we considered all the unsafe values described in RQ1.4. For each unsafe value, we provided a descriptive sentence (see Table 11) to be selected by the participants from a checkbox list. Participants could select more than one option and we provided the option *None of the above*, for participants who did not find the set of provided answers to be satisfactory. Each questionnaire was introduced by a short description of SAFE and a detailed description of the task to perform. Further, we provided an example answer from a different case that is simple to understand (classification of animal pictures).

Data collection was automated using Lime Survey<sup>6</sup> and its link was sent to the participants by email. Figure 11 shows an example question provided to the participants (in this case, the images belong to one root cause cluster generated for the GD DNN). An example of a questionnaire sent to the participants can be found in our replicability package.

<sup>6</sup><https://ulsurvey.uni.lu/>

Table 11. The descriptive sentence used in the survey for each unsafe value

Parameter	Descriptive sentence	Unsafe value
Headpose-Y	The face is straight forward	0°
Headpose-Y	The face is partially not visible because it is inclined to the top	74.17°
Headpose-Y	The face is partially not visible because it is inclined to the bottom	-88.10°
Headpose-X	The face is partially not visible because it is turned to the left side of the image	-28.88°
Headpose-X	The face is partially not visible because it is turned to the right side of the image	21.35°
Headpose-X	The face is turned between the middle and the left side of the image	-10°
Headpose-X	The face is turned between the middle and the right side of the image	10°
Headpose-Y	The face is turned between the centre and the top of the image	10°
Headpose-Y	The face is looking between the centre and the bottom of the image	-10°
Openness	The eyes are abnormally open	>64°
DistToCenter	The eyes are looking middle centre	<11.5°
Gaze Angle	The eyes are looking between the top left and the middle left of the image	22.5°
Gaze Angle	The eyes are looking between the top centre and the top left of the image	67.5°
Gaze Angle	The eyes are looking between top right and top centre	112.5°
Gaze Angle	The eyes are looking between the middle right and the top right of the image	157.5°
Gaze Angle	The eyes are looking between the bottom right and the middle right of the image	202.5°
Gaze Angle	The eyes are looking between the bottom centre and the bottom right of the image	247.5°
Gaze Angle	The eyes are looking between bottom left and bottom centre	292.5°
Gaze Angle	The eyes are looking between the middle left and the bottom left of the image	337.5°
PupilToTop	The pupil is mostly above the eyelid	<-16°
PupilToBottom	The pupil is mostly under the eyelid	>-16°

For each set of answers (i.e., the ones provided with 5, 10, or 15 images), we counted the number of answers that matched our ground truth. The number of images required to determine the root causes of a DNN error is the minimal number leading to a high percentage of correct answers.

To summarize, based on the experimental design above, we prevented learning effects from one question to the next. We ensured that, for the same case study, the selected cluster and images were different for each participant and selected randomly, to avoid any form of systematic bias.

Table 12. Percentage of correct responses by inspecting 5, 10 and 15 images for each case study subject, based on the questionnaire study.

Case Study Subjects	Correct responses		
	5 images inspected	10 images inspected	15 images inspected
GD	80%	88%	67%
OC	83%	83%	86%
HPD	100%	83%	67%
Overall	89%	84%	74%

*Results.* Table 12 presents the results obtained from analyzing the questionnaire data. Overall, we notice that when looking at 5, 10, and 15 images, 89% 84%, and 74% of the participants found the correct commonality in a cluster, respectively. Given that participants performed that type of task for the first time, with limited initial training, we can consider 89% to be a good result and a lower bound of what experienced practitioners would achieve. However, surprisingly, a larger number of inspected images does not improve results. On the contrary, with a larger number of images being inspected (e.g., 15 images), the percentage of correct answers tends to drop. This result may be due to the fact that a larger set of images leads to a higher cognitive load and is also more likely to include noisy images (i.e., images that do not present the same commonalities as most of the other images in the cluster). In the presence of noisy images, a user who looks for a commonality across all the images may identify characteristics that are not a correct explanation for the DNN error.

Since we are dealing with proportions, we performed a Fisher exact test to determine if the differences observed between the pairs *5 images VS 10 images* and *5 images VS 15 images* are significant. The obtained p-values (i.e., 1 for '5 vs 10' and 0.4 for '5 vs 15') indicate that differences are not significant, which may be due in part to the small number of participants (19). However, our results clearly show that the inspection of five images does not lead to worse results than the inspection of a higher number of images, thus justifying our assumption in RQ1.1 (i.e., engineers inspect five images per cluster).

GD and OC lead to similar results, with 80% (GD) and 83% (OC) of the participants who inspected five images providing a correct answer. HPD leads to better results; indeed, 100% of the participants found the correct commonality by inspecting five images. Such difference between HPD and the other two cases is likely due to the fact that the simulator used to generate HPD images is more realistic (i.e., generates a whole face), while the simulator used for GD and OC simply generates eye bulbs and part of the forehead (see Figure 7), thus resulting in images that are more complicated to quickly understand by participants who are not familiar with the application domain. Despite such differences, which are to be expected across case studies, the generic trend is consistent: five images seem to be sufficient and not worse than larger sets of images. Therefore, we conclude that inspecting five images per cluster is an acceptable choice regardless of the case study DNN.

#### 4.2.6 RQ2. Does SAFE enable engineers to more effectively and efficiently retrain a DNN when compared with HUDD and baselines approaches?

*Design and measurements.* In this experiment, we investigate whether SAFE significantly improves the accuracy due to retraining the DNN, thanks to its selection method accounting for the shape of clusters. We compare these improvements to HUDD and two baselines, namely BL1 and BL2, which are depicted in Figure 12 and explained in the following:

*BL1*: In this baseline, we use the error-inducing images in the improvement set as an unsafe set. Precisely, we label<sup>7</sup> a random subset of the improvement set and execute the DNN to identify error-inducing images. As for HUDD, this selected unsafe set is augmented by applying bootstrap resampling (i.e., replicating samples in the unsafe set) in order to have a sufficiently large number of unsafe images to improve the DNN.

*BL2*: This baseline consists of randomly selecting a set of images from the improvement set, labeling them to obtain a *labeled selected improvement set*, and using them for retraining.

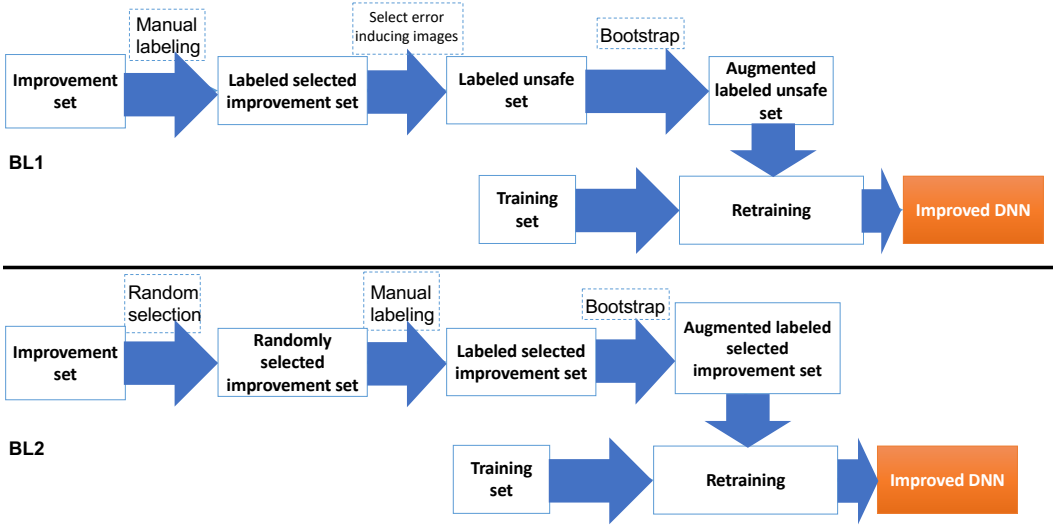


Fig. 12. Process of the two baselines used to compare SAFE

We rely on the same settings and environment to run the experiments for the four different retraining strategies (i.e., SAFE, HUDD, BL1, and BL2). These experiments are repeated ten times to account for randomness.

To retrain DNNs, we rely on the approach described in Section 3.6. For fair comparisons with HUDD [12], BL1, and BL2, we configure bootstrap resampling to generate an *augmented labeled unsafe set* and an *augmented labeled selected improvement set* with the same size as the *balanced labeled unsafe set* for HUDD (see Figure 2).

We compute the accuracy of the retrained models on the test sets and compare the accuracy improvement obtained by SAFE with those obtained by HUDD and the baselines. For this experiment, we consider the case study subjects presented in Table 1.

The improvement sets for GD and OC have been generated through additional executions of UnityEyes. For HPD and FLD, they 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. We selected images of the original datasets not used for the training and test sets for the other cases.

<sup>7</sup>For our experiments, labeling comes for free because we either derive images using a simulator or we rely on available datasets. However, labeling comes at a high cost in industrial practice, where new images are collected from the field.

Table 13. RQ2: Unsafe set size and the accuracy improvement of SAFE compared to HUDD

DNN	Original Model	SAFE			HUDD			BL1	BL2
		Unsafe set size	Accuracy	Improvement over original model	Unsafe set size	Accuracy	Improvement over original model	Accuracy	Accuracy
GD	95.95%	1648	96.74%	<b>+0.79</b>	1615	96.23%	+0.28	95.23%	95.80%
OC	88.03%	153	96.29%	<b>+8.2</b>	160	94.41%	+6.38	91.65%	92.33%
HPD	44.07%	438	69.58%	<b>+25.51</b>	481	68.13%	+24.06	66.73%	66.30%
FLD	44.99%	659	79.17%	<b>+34.18</b>	502	75.23%	+30.24	72.02%	73.83%
TS	81.65%	597	93.47%	<b>+11.82</b>	704	93.03%	+11.38	92.63%	92.73%
OD	84.12%	212	97.14%	<b>+13.02</b>	258	97.04%	+12.92	97.04%	96.67%

Table 14. RQ2: p-values and VDA values when comparing SAFE to HUDD and the baselines

	GD			OC			TS			OD			HPD			FLD		
p-value	0.003	0.003	0.001	0.000	0.000	0.000	0.007	0.003	0.001	0.22	0.002	0.04	0.003	0.000	0.000	0.000	0.000	0.000
VDA	0.88	0.88	0.9	1	1	1	0.85	0.96	1	0.66	0.9	0.77	0.85	1	1	1	1	1

*Results.* Table 13 provides an accuracy comparison between SAFE, HUDD, and the two baselines on the different case study DNNs. It also provides the size of the unsafe sets (number of images) selected by each method.

For SAFE, improvements in accuracy over the original model varies from 0.79 to 34.18, compared to lower ranges for HUDD (from 0.28% to 30.24%), BL1 (from -0.18% to 27.03%) and BL2 (-0.15% to 28.84%). Across case study DNNs, SAFE clearly and systematically yields better accuracy results compared to HUDD and the baselines, though to varying extents. Note that unsafe set sizes are comparable across improvement strategies and that differences in accuracy are therefore due to the strategy adopted for selecting unsafe images.

The lower improvement shown by the baselines can be explained by the fact they do not rely on root cause clusters to select images for the unsafe set, a strategy that appears to be beneficial in our context. SAFE always yields higher improvement than that of HUDD. This is because HUDD relies on the cluster's centroids to select images for the unsafe set without accounting for the cluster's shape. In contrast, SAFE relies on core points to choose the images for the unsafe set. This enables the handling of clusters with arbitrary shapes as the selected images take the shape of the cluster, as explained in Section 3.6.

Regarding SAFE results, we notice three types of improvement explained in the following:

For GD, the improvement is only +0.79. This is because the accuracy of the original model for this case study subject was already very high (95.95%), with limited room for improvement.

For HPD and FLD, the improvement in accuracy is +26.16 and +34.18, respectively. The original models for these two case study subjects had low accuracy and we therefore expected a large improvement. In fact, HPD and FLD represent two cases where retraining was very much needed.

For OC, TS and OD, the improvement in accuracy is +8.2, +11.82 and +13.18, respectively. Though this improvement is moderate compared to the previous case study DNNs, it is still significant given the original models' relatively high accuracy. After retraining, SAFE achieved high accuracy for these case study subjects with 96.29%, 93.47%, and 97.30%, respectively.

We report the significance of these results in Table 14, including the values of the Vargha and Delaney's  $\hat{A}_{12}$  effect size and the p-values resulting from performing a Mann-Whitney U-test



Table 15. Execution time of the feature extraction of the improvement sets and the memory allocations of these features for SAFE. Compared to the execution time of the generation of the heatmaps and their memory allocation.

Subject	Execution time (s)		Memory allocation (Mb)	
	SAFE	HUDD	SAFE	HUDD
GD	176	3,920	74.2	78,641
OC	160	958	4.1	3,551
HPD	161	1,294	1.6	8,839
FLD	210	1,883	0.82	11,981
OD	149	1,059	13.2	5,989
TS	152	2,335	1.6	16,744

between the accuracy of SAFE and other improvement strategies. Recall that we run each strategy 10 times.

Typically, an  $\hat{A}_{12}$  effect size above 0.56 is considered significant with higher thresholds for medium (0.64) and large (0.71) effects, thus suggesting the effect sizes between SAFE and other strategies are large across case study DNNs, in all but one case.

We notice in Table 14 that the p-values when comparing SAFE to the baselines are always below 0.05. As for HUDD, the p-values are lower than 0.05 in 5 out of 6 case study subjects. This implies that in most cases, the null hypothesis is rejected.

#### 4.2.7 RQ3. Does SAFE provide time and memory savings compared to HUDD?

*Design and measurements.* As mentioned in the previous Section, SAFE proved its effectiveness in retraining the different case study subjects as it obtained significant improvements over the original models. SAFE not only performs better than HUDD but also, and perhaps more significantly, provides very high time and memory savings. This research question investigates whether SAFE, with its black-box nature, offers significant time and memory saving.

We compare the time required by SAFE and HUDD to perform their most expensive tasks, which are the time required to extract the features by SAFE and to generate the heatmaps by HUDD; we do not report the time required to perform the other steps of the two approaches because these steps are either shared or do not have any practical impact on performance (i.e., they took few seconds in our experiments). We also compare the memory allocation of the features and the heatmaps, which are the data types processed only by SAFE and HUDD, respectively.

*Results.* Table 15 provide our results. We observe a large time and memory saving for SAFE compared to HUDD. Such performance considerations have significant practical implications. For example, we can observe that SAFE requires, in the worst case, only 3.5 minutes to generate RCCs and the unsafe set to be used for retraining. HUDD, instead, requires 65 minutes to achieve the same objective. Such difference has a huge impact on the practicality of the approach as, for SAFE, the analysis of RCCs can be performed shortly after observing DNN failures. HUDD may require up to one hour to do the same. Such execution time savings allow engineers to conduct DNN safety analysis and improvements in a much shorter time. Memory savings also have significant implications since it prevents the need for expensive hardware to perform such analysis.

The explanation for the above results is that SAFE is using extracted features to represent the images instead of heatmaps. Feature extraction is less costly in time and memory than the computation of heatmaps. Indeed, the distance computed on features is less computationally complex than the one calculated on heatmaps.

Heatmaps also take a great deal of memory for storage. HUDD generates heatmaps for each layer. For instance, the heatmap for the eighth layer of AlexNet has a size of  $169 \times 256$  (convolution

layer), while the heatmap for the tenth layer has a size of  $4096 \times C$ . With this architecture, HUDD will generate eight heatmaps of size  $169 \times 256$  and one heatmap of size  $4096 \times C$  for every image in the dataset. In contrast, for SAFE, each image is represented by a  $1 \times 256$  matrix (256 features for each image).

### 4.3 Threats to validity

We discuss internal, conclusion, construct, and external validity according to conventional practices [76].

*4.3.1 Internal validity.* A possible internal threat is the use of the feature extraction method on which we rely, which could negatively affect our results if inadequate. Indeed, clustering relies on the similarity computed in terms of the extracted features. To mitigate this threat, we have checked that some of the features extracted by our method are consistent within clusters, by visually inspecting them. Indeed, such features contain enough information on the images if the clusters are visually consistent, thus demonstrating that the features extraction method worked.

*4.3.2 External validity.* The selection of the case study DNNs could be a threat to validity. This paper alleviates this issue by using six datasets with diverse complexity. Four subject DNNs out of six implement tasks motivated by IEE business needs that address problems that are quite common in the automotive industry. Also, the simulators used in our experiments, though being related to IEE in-car sensing business cases, vary in terms of characteristics; indeed, they range from high-fidelity simulation of specific body parts (in our case, the human eye) to whole human body simulations (we crop the face) with lower fidelity. In our experiments we test DNNs that process cropped images (e.g., human's head, traffic signs). Cropping, which a DNN can perform, limits the number of features appearing in the images to be processed, thus potentially simplifying the task to be performed by SAFE. However, cropping was justified in our context by the expected inputs of our subject DNNs. Finally, although we focus on data sets related to in-car sensing, we believe that SAFE will perform well with other data sets since the VGG model used for the feature extraction was pre-trained on Image-Net, which means that the model can capture features related to 1000 classes, including humans, animals, and objects. In the future, we aim to extend our work to include subjects from different domains (e.g., different types of classification tasks with non-cropped images).

Another threat to the generalizability of our results is the dependence on ImageNet. SAFE relies on VGG16 to extract features from images. VGG16 was pre-trained on ImageNet, which is a large image database. Therefore, SAFE is expected to work better with images containing objects recognized by ImageNet. However, we believe that this characteristic does not affect the practical applicability of SAFE in the automotive context since the ImageNet VGG recognizes 1000 different objects, including objects belonging to the automotive scenery; these objects include cars, faces, and eyes, for example <sup>8</sup>.

*4.3.3 Conclusion validity.* To avoid violating parametric assumptions in our statistical analysis, we rely on a non-parametric test and effect size measure (i.e., Mann Whitney U-test, the Vargha and Delaney's  $\hat{A}_{12}$  statistics, and the one-sample Wilcoxon signed rank test, respectively) to evaluate the statistical and practical significance of differences in results. We report both p-values and effect sizes.

Due to the stochastic nature of SAFE (e.g., DNN retraining), the experiments conducted with the SAFE method were executed over ten runs. We reported the descriptive statistics of those runs and discussed the statistical significance and effect size of differences across methods.

<sup>8</sup>The full list of classes is available at <https://gist.github.com/yrevar/942d3a0ac09ec9e5eb3a>

**4.3.4 Construct validity.** The constructs considered in our work are effectiveness and cost. Effectiveness is measured through complementary indicators, which include (1) within-cluster variance reduction for at least one parameter (for RQ1.2), (2) average values being close to unsafe values for parameters with high within-cluster variance reduction (for RQ1.3), (3) coverage of plausible causes of errors represented by unsafe values (for RQ1.4), (4) DNN accuracy improvement (for RQ2). Although the effectiveness regarding the analysis of root causes (i.e., RQ1.2 to RQ1.4) might be evaluated with user studies, such evaluation might be biased by the background and experience of the selected pool of users, which shall also be sufficiently large in number. In our context, end-users are engineers with background in safety analysis (e.g., to determine if an input is realistic or if a DNN error may lead to a hazard) and machine learning. However, since safety experts are generally not trained to use DNNs whereas DNN experts (e.g., recently graduated students) are typically not safety experts, it would be difficult to select a large enough set of users for the study. For this reason, we preferred to rely on reflective indicators based on the information provided by simulators, thus enabling an objective evaluation. Moreover, the quality and usefulness of our results have been confirmed by experts at our industry partner, IEE Sensing, over multiple technical and management meetings. These experts included researchers with a Ph.D. in mathematics and machine learning who develop safety-software components, safety engineers integrating DNN-based components, and chief technology officers. To measure the effectiveness of DNN retraining, we relied on improvements in accuracy, which is common practice.

Concerning cost, we discussed the feasibility of root cause analysis by reporting the number of clusters generated by the approach and the number of images that are sufficient to determine commonalities across images, based on our experience and that of our industry partners. We also discussed the cost of DNN retraining by reporting the time required for retraining, which affects the feasibility of the approach in practice, and memory allocation costs, which affect hardware requirements.

Another threat is with RQ1.3 (Section 4.2.3) where we systematically select unsafe parameters for each case study to evaluate the clusters. This can prevent us from identifying other reasons for failures if, for any reason, we miss some of these parameters. However, whether this is the case or not, this does not prevent the identification of clusters and missing parameters would then lead to clusters without clear root causes, something we have not observed in our experiments.

## 5 RELATED WORK

Most of the DNN testing and analysis approaches are summarized in recent surveys [24, 81]. However, no survey on the automated debugging and retraining of DNNs has been proposed to date.

AUTOTRAINER is a DNN monitoring and auto-repairing system [82]. It monitors the training status of the model and automatically fixes it (by retraining) once a problem is detected. AUTOTRAINER can efficiently detect and repair five targeted training problems (i.e., vanishing gradient, exploding gradient, dying ReLU, oscillating loss, and slow convergence). Despite its effectiveness in detecting these problems, AUTOTRAINER cannot explain certain misclassifications since it cannot detect a root cause of the error that is not defined a priori.

AI-Lancet optimizes deep learning models by locating the error-inducing (EI) neurons and fixing them using either neuron-flip or neuron-fine-tuning methods [84]. It starts by revealing the EI regions in the input sample and then extracts the EI features activated by the EI regions of the input. Finally, the EI neurons can be located with the guidance of the EI features. Unlike SAFE, AI-Lancet requires the modification and the retraining of the DNN to find the erroneous neurons. Another limitation is that it does not explain the root causes of errors. Instead, it attempts to fix them by fine-tuning or flipping neurons.

MODE [44] evaluates each layer to identify buggy neurons and further generates fixed-size batches of images for retraining. However, setting such size can be difficult when dealing with datasets with complex features. Also, in the improvement set selection step, MODE requires the modification and the retraining of the DNN. In SAFE, the unsafe set is selected automatically. Another difference between MODE and SAFE is that the former cannot detect the root cause of a DNN error, which is one of the most important features of the latter. Further, HUDD outperformed MODE based on OD (the only usable dataset to compare since MODE authors did not provide an implementation). SAFE also outperformed both HUDD and MODE based on the OD dataset (97.14% compared to 97.04% for HUDD and 89% for MODE).

Apricot [80] is a two-phases approach. The first phase is weight adjustment, where for each failing input  $x$ , it adjusts the weights of the model by running different DNN's on different subsets of the training and test sets. The second phase is retraining, where the DNN is retrained using the entire training set with the new adjusted weights. In addition to the low accuracy improvement shown by this method (less than 2%), it also requires the manipulation of the DNN (adjusting the weights).

Kim et al. [33] use Surprise Adequacy (SA) to guide the selection of newly collected inputs to be added to the base training dataset for the retraining of a DNN-based semantic segmentation module for autonomous driving in the automotive industry. SA measures how *Surprising* an input is to the DNN (i.e., how different this input is from the ones the network has already seen). The main limitation of this method is that it does not explain the DNN failures.

RobOT (Robustness-Oriented Testing) iteratively improves the robustness of a DNN model by generating adversarial inputs that can be used to test the model and retrain it [74]. The test generation is driven by the first-order loss, which measures the loss achieved by the input generated from a given seed. RobOT outperforms related approaches [13, 37]. Different from SAFE, RobOT aims to improve robustness rather than DNN accuracy; also, it does not include strategies to provide explanations.

Some DNN testing approaches can provide explanations for the input regions in which DNN errors are observed. For example, Abdessalem et al. [1] rely on evolutionary search to drive the generation of test inputs using simulators; to maximize effectiveness, decision trees are used to learn, during search, the portions of the input space that are less safe and, therefore, should be targeted by testing. The decision tree leaves that characterize such unsafe portions are then presented to the end-users. Recent work further demonstrates the effectiveness of decision trees to characterize the input space, based on the results obtained during simulator-based testing [20]. DeepHyperion [86] relies on a metaheuristic search algorithm to configure a generative model (e.g., a simulator) to generate test inputs towards specific dimension of the input space (e.g., image orientation); then, it provides to the end-user a set of feature maps that visualize the degree of accuracy obtained for varying values in pairs of dimensions. The main limitation of these DNN testing approaches is that, different from SAFE, they can provide explanations only for inputs generated with simulators, not for real-world inputs.

Further, most of the previously mentioned approaches rely on white-box techniques, which means that the modification of the DNN and a specific set of its parameters is required. SAFE overcomes these limitations by proposing a black-box approach based on a pre-trained feature extraction model trained on the ImageNet dataset. The pre-trained model is used as-is to extract the features and then select an unsafe set for retraining.

Another advantage of SAFE over the other methods is in helping detect error root causes in the DNN, using a clustering algorithm. As discussed, this is required in the context of safety analysis. Further, every cluster representing a root cause can be used to select images to improve the model by retraining it and thus making it more robust to any targeted root cause.

Both SAFE and HUDD identify different situations in which image-processing DNNs are likely to trigger an erroneous result. However, the former fares better than HUDD with large time and memory savings due to the use of extracted features to represent images instead of heatmaps. Features require less time to extract and less memory to store.

Another state-of-the-art approach for the generation of explanations is the Anchor algorithm, which derives decision rules (called *anchor explanations*) that sufficiently tie a prediction locally [56]. Changes to the rest of the feature values do not matter, i.e., similar instances covered by the same anchor have the same prediction outcome. The Anchor algorithm is applied on tabular images and textual datasets. The Anchor algorithm constructs an explanation rule iteratively by interacting with the model to be explained. Each iteration alters the values associated with one input feature until it identifies a range within which the accuracy is above a given threshold. Though designed for textual datasets, the Anchor algorithm may provide decision rules that can be easily generalized to multiple inputs (e.g., the ones that match the same rule and lead to the same result); for image inputs, Anchor simply emphasizes the image chunk that is sufficient for the classifier to make the prediction. Therefore, Anchor does not help engineers in analyzing large input datasets because it requires all the chunks belonging to each input image to be visualized. SAFE, instead, enables engineers to efficiently identify root causes from large sets of error-inducing images; further, it automatically retrains the DNN, a task not supported by Anchor.

## 6 CONCLUSION

In this paper, we presented SAFE, a new black-box approach that automatically identifies the different situations in which a DNN is more likely to fail, without requiring any modification to the DNN or access to its internal information. Similar to our previous white-box approach (i.e., HUDD), SAFE characterizes such situations by generating clusters of images that likely lead to a DNN error because of the same underlying reason. We refer to these clusters as root cause clusters. Differently from HUDD, SAFE is based on a pre-trained model to extract features from error-inducing images. These features replace the heatmaps used by HUDD to generate the root cause clusters. Also, SAFE uses a density-based clustering algorithm to generate arbitrary-shaped clusters.

SAFE uses a new method to select an unsafe set for retraining based on the cluster's core points. More specifically, SAFE selects images close to each cluster's core points. These images are manually labeled and used to augment the training set to improve the DNN through retraining.

Empirical results show that SAFE derives root cause clusters that can effectively help engineers determine the root causes for DNN errors. Indeed, the number of generated clusters is low, thus making the visual inspection of a few representative images for each cluster feasible. They include images with similar characteristics that are likely related to the cause of the error. Further, for our case study subjects, the generated clusters capture all the possible causes of errors. Compared to HUDD, SAFE generates clusters with more similar characteristics covering a larger set of error causes. Moreover, the DNNs retrained by SAFE achieve a higher accuracy than that obtained with HUDD and baseline approaches.

Besides the benefits described above, SAFE also saves large amounts of execution time and memory due to its black-box nature and the reliance on the extracted features instead of heatmaps, thus making it more usable in practical contexts.

## ACKNOWLEDGMENTS

This project has received funding from IEE Luxembourg, Luxembourg's National Research Fund (FNR) under grant BRIDGES2020/IS/14711346/FUNTASY, and NSERC of Canada under the Discovery and CRC programs. Authors would like to thank Thomas Stifter from IEE for his 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] Raja Ben Abdesslem, Shiva Nejati, Lionel C Briand, and Thomas Stifter. 2018. Testing vision-based control systems using learnable evolutionary algorithms. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*. IEEE, 1016–1026.
- [2] Adel Alaeddini and Ibrahim Dogan. 2011. Using Bayesian networks for root cause analysis in statistical process control. *Expert Systems with Applications* 38, 9 (2011), 11230–11243.
- [3] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. 2017. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*. Ieee, 1–6.
- [4] Maximilian Alber, Sebastian Lapuschkin, Philipp Seegerer, Miriam Hägele, Kristof T. Schütt, Grégoire Montavon, Wojciech Samek, Klaus-Robert Müller, Sven Dähne, and Pieter-Jan Kindermans. 2019. iNNvestigate Neural Networks! *Journal of Machine Learning Research* 20, 93 (2019), 1–8. <http://jmlr.org/papers/v20/18-540.html>
- [5] Authors of this paper. 2022. SAFE: toolset and replicability package. <https://zenodo.org/record/6619279>
- [6] Purnima Bholowalia and Arvind Kumar. 2014. EBK-means: A clustering technique based on elbow method and k-means in WSN. *International Journal of Computer Applications* 105, 9 (2014).
- [7] Piotr Dabkowski and Yarin Gal. 2017. Real Time Image Saliency for Black Box Classifiers. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6970–6979.
- [8] David L Davies and Donald W Bouldin. 1979. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence* 2 (1979), 224–227.
- [9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (June 2019), 4171–4186. <https://doi.org/10.18653/v1/N19-1423>
- [10] Nassima Dif, Mohammed Oualid Attaoui, Zakaria Elberrichi, Mustapha Lebbah, and Hanene Azzag. 2021. Transfer learning from synthetic labels for histopathological images classification. *Applied Intelligence* (2021), 1–20.
- [11] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. 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 (KDD'96)*. AAAI Press, 226–231.
- [12] Hazem Fahmy, Fabrizio Pastore, Mojtaba Bagherzadeh, and Lionel Briand. 2021. Supporting Deep Neural Network Safety Analysis and Retraining Through Heatmap-Based Unsupervised Learning. *IEEE Transactions on Reliability* (2021), 1–17. <https://doi.org/10.1109/TR.2021.3074750>
- [13] Yang Feng, Qingkai Shi, Xinyu Gao, Jun Wan, Chunrong Fang, and Zhenyu Chen. 2020. DeepGini: Prioritizing Massive Tests to Enhance the Robustness of Deep Neural Networks. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2020)*. Association for Computing Machinery, New York, NY, USA, 177–188. <https://doi.org/10.1145/3395363.3397357>
- [14] Hao Fu, Shanjiang Tang, Ce Yu, Yusen Li, Jizhou Sun, and Yanjie Liu. 2021. DVQShare: An Analytics System for DNN-based Video Queries. In *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE, 166–175.
- [15] Rafael Garcia, Alexandru C. Telea, Bruno Castro da Silva, Jim Torresen, and Joao Luiz Dihl Comba. 2018. A task-and-technique centered survey on visual analytics for deep learning model engineering. *Computers and Graphics* 77 (2018), 30 – 49. <https://doi.org/10.1016/j.cag.2018.09.018>
- [16] Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. 2018. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*. IEEE, 80–89.
- [17] Ana Gómez-Andrades, Pablo Munoz, Inmaculada Serrano, and Raquel Barco. 2015. Automatic root cause analysis for LTE networks based on unsupervised techniques. *IEEE Transactions on Vehicular Technology* 65, 4 (2015), 2369–2386.
- [18] Alexander N Gorban and Andrei Y Zinovyev. 2010. Principal graphs and manifolds. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI Global, 28–59.
- [19] Gábor Gosztolya, Róbert Busa-Fekete, Tamás Grósz, and László Tóth. 2017. DNN-Based Feature Extraction and Classifier Combination for Child-Directed Speech, Cold and Snoring Identification. In *Proceeding of Interspeech 2017*. International Speech Communication Association (ISCA), 3522–3526. <https://doi.org/10.21437/Interspeech.2017-905>
- [20] Fitash Ul Haq, Donghwan Shin, Lionel C. Briand, Thomas Stifter, and Jun Wang. 2021. Automatic Test Suite Generation for Key-Points Detection DNNs Using Many-Objective Search (Experience Paper). In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2021)*. Association for Computing Machinery, New York, NY, USA, 91–102. <https://doi.org/10.1145/3460319.3464802>

- [21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
- [22] Zhenzhen He, Yihai He, and Yi Wei. 2016. Big data oriented root cause identification approach based on PCA and SVM for product infant failure. In *2016 Prognostics and System Health Management Conference (PHM-Chengdu)*. IEEE, 1–5.
- [23] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR* abs/1704.04861 (2017). arXiv:1704.04861
- [24] Xiaowei Huang, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu, and Xinpeng Yi. 2020. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review* 37 (2020), 100270.
- [25] Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification* 2, 1 (1985), 193–218.
- [26] Nargiz Humbatova, Gunel Jahangirova, Gabriele Bavota, Vincenzo Riccio, Andrea Stocco, and Paolo Tonella. 2020. Taxonomy of Real Faults in Deep Learning Systems. In *Proceedings of the 42nd International Conference on Software Engineering*. Association for Computing Machinery, New York, NY, USA, 10.
- [27] IEE. 2020. IEE Sensing solutions. [www.iee.lu](http://www.iee.lu).
- [28] INI. 2020. TRaffic Sign Dataset. <http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset>
- [29] International Organization for Standardization. 2020. ISO, ISO-24765-2017, Systems and software engineering - Vocabulary.
- [30] International Organization for Standardization. 2020. ISO, ISO26262-1:2018, Road vehicles: Functional safety.
- [31] Jeya Vikranth Jeyakumar, Joseph Noor, Yu-Hsi Cheng, Luis Garcia, and Mani Srivastava. 2020. How Can I Explain This to You? An Empirical Study of Deep Neural Network Explanation Methods. 33 (2020), 4211–4222.
- [32] Benish Kabir, Pamir, Ashraf Ullah, Shoaib Munawar, Muhammad Asif, and Nadeem Javaid. 2021. Detection of Non-Technical Losses Using MLP-GRU Based Neural Network to Secure Smart Grids. In *Complex, Intelligent and Software Intensive Systems*, Leonard Barolli, Kangbin Yim, and Tomoya Enokido (Eds.). Springer International Publishing, Cham, 383–394.
- [33] Jinhan Kim, Jeongil Ju, Robert Feldt, and Shin Yoo. 2020. Reducing dnn labelling cost using surprise adequacy: An industrial case study for autonomous driving. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1466–1476.
- [34] Hans-Peter Kriegel, Peer Kröger, Jörg Sander, and Arthur Zimek. 2011. Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1, 3 (2011), 231–240.
- [35] A. Krizhevsky and G. Hinton. 2009. *Learning multiple layers of features from tiny images*. Technical Report. Department of Computer Science, University of Toronto.
- [36] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* 60, 6 (May 2017), 84–90. <https://doi.org/10.1145/3065386>
- [37] Seokhyun Lee, Sooyoung Cha, Dain Lee, and Hakjoo Oh. 2020. Effective White-Box Testing of Deep Neural Networks with Adaptive Neuron-Selection Strategy. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2020)*. Association for Computing Machinery, New York, NY, USA, 165–176. <https://doi.org/10.1145/3395363.3397346>
- [38] Yaguo Lei, Feng Jia, Jing Lin, Saibo Xing, and Steven X Ding. 2016. An intelligent fault diagnosis method using unsupervised feature learning towards mechanical big data. *IEEE Transactions on Industrial Electronics* 63, 5 (2016), 3137–3147.
- [39] Zhong Li, Minxue Pan, Tian Zhang, and Xuandong Li. 2021. Testing DNN-based Autonomous Driving Systems under Critical Environmental Conditions. In *International Conference on Machine Learning*. PMLR, 6471–6482.
- [40] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *Computer Vision – ECCV 2014*. Springer International Publishing, Cham, 740–755.
- [41] Johan Linaker, Sardar Muhammad Sulaman, Martin Höst, and Rafael Maiani de Mello. 2015. Guidelines for conducting surveys in software engineering v. 1.1. *Lund University* (2015).
- [42] Z. Liu, P. Luo, X. Wang, and X. Tang. 2015. Deep Learning Face Attributes in the Wild. In *2015 IEEE International Conference on Computer Vision (ICCV)*. 3730–3738.
- [43] Mohamed Loey, Gunasekaran Manogaran, Mohamed Hamed N Taha, and Nour Eldeen M Khalifa. 2021. A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic. *Measurement* 167 (2021), 108288.
- [44] Shiqing Ma, Yingqi Liu, Wen-Chuan Lee, Xiangyu Zhang, and Ananth Grama. 2018. 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 (ESEC/FSE 2018)*. ACM, New York, NY, USA, 175–186. <https://doi.org/10.1145/3236024.3236082>

- [45] Levoy Marc, Marc Levoy, Rusinkiewicz Szymon, Weyrich Tim, Pfister Hanspeter, Amenta Nina, Wu Jianhua, Barthe Loïc, Zwicker Matthias, Kobbelt Leif, et al. 2007. 2-THE EARLY HISTORY OF POINT-BASED GRAPHICS. In *Point-Based Graphics*. Elsevier, 8–16.
- [46] Leland McInnes, John Healy, and Steve Astels. 2017. hdbscan: Hierarchical density based clustering. *Journal of Open Source Software* 2, 11 (2017), 205.
- [47] Leland McInnes, John Healy, and James Melville. 2020. UMAP: uniform manifold approximation and projection for dimension reduction. (2020).
- [48] Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus Robert Müller. 2019. *Layer-Wise Relevance Propagation: An Overview*. Springer International Publishing, Cham, 193–209. [https://doi.org/10.1007/978-3-030-28954-6\\_10](https://doi.org/10.1007/978-3-030-28954-6_10)
- [49] Rajaditya Mukherjee, Qingyang Li, Zhili Chen, Shicheng Chu, and Huamin Wang. 2018. Neuraldrop: Dnn-based simulation of small-scale liquid flows on solids. *arXiv preprint arXiv:1811.02517* (2018).
- [50] Alejandro Newell, Kaiyu Yang, and Jia Deng. 2016. Stacked Hourglass Networks for Human Pose Estimation. In *Computer Vision – ECCV 2016*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (Eds.). Springer International Publishing, Cham, 483–499.
- [51] Renjian Pan, Zhaobo Zhang, Xin Li, Krishnendu Chakrabarty, and Xinli Gu. 2021. Unsupervised Two-Stage Root-Cause Analysis for Integrated Systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2021).
- [52] Karl Pearson. 1901. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science* 2, 11 (1901), 559–572.
- [53] Vitali Petsiuk, Abir Das, and Kate Saenko. 2018. RISE: Randomized Input Sampling for Explanation of Black-box Models. In *Proceedings of the British Machine Vision Conference (BMVC)*.
- [54] PyTorch. 2020. PyTorch DNN framework. <https://pytorch.org>
- [55] Nadia Rahmah and Imas Sukaesih Sitanggang. 2016. Determination of optimal epsilon (eps) value on dbscan algorithm to clustering data on peatland hotspots in sumatra. In *IOP conference series: earth and environmental science*, Vol. 31. IOP Publishing, 012012.
- [56] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.
- [57] Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* 20 (1987), 53–65.
- [58] Madona B Sahaai et al. 2021. Brain Tumor Detection using DNN Algorithm. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)* 12, 11 (2021), 3338–3345.
- [59] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. 2017. DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. *ACM Trans. Database Syst.* 42, 3, Article 19 (jul 2017), 21 pages. <https://doi.org/10.1145/3068335>
- [60] SciPy. 2020. Python framework for mathematics, science, and engineering. <https://scipy.org/>
- [61] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. 2017. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*. 618–626. <https://doi.org/10.1109/ICCV.2017.74>
- [62] Jonathon Shlens. 2014. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100* (2014).
- [63] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
- [64] Sandeep Sony, Kyle Dunphy, Ayan Sadhu, and Miriam Capretz. 2021. A systematic review of convolutional neural network-based structural condition assessment techniques. *Engineering Structures* 226 (2021), 111347.
- [65] J.T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. 2015. Striving for Simplicity: The All Convolutional Net. In *ICLR (workshop track)*.
- [66] Stanford Vision Lab. 2022. ImageNet, image database organized according to the WordNet hierarchy. <https://www.image-net.org>.
- [67] Alexander Strehl and Joydeep Ghosh. 2002. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research* 3, Dec (2002), 583–617.
- [68] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.
- [69] Muhammed Talo. 2019. Automated classification of histopathology images using transfer learning. *Artificial Intelligence in Medicine* 101 (2019), 101743.
- [70] Yuchi Tian. 2021. *Detect and Repair Errors for DNN-based Software*. Ph.D. Dissertation. Columbia University.
- [71] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. 2018. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*. 303–314.



- [72] G Vallathan, A John, Chandrasegar Thirumalai, SenthilKumar Mohan, Gautam Srivastava, and Jerry Chun-Wei Lin. 2021. Suspicious activity detection using deep learning in secure assisted living IoT environments. *The Journal of Supercomputing* 77, 4 (2021), 3242–3260.
- [73] Zitong Wan, Rui Yang, Mengjie Huang, Nianyin Zeng, and Xiaohui Liu. 2021. A review on transfer learning in EEG signal analysis. *Neurocomputing* 421 (2021), 1–14.
- [74] Jingyi Wang, Jialuo Chen, Youcheng Sun, Xingjun Ma, Dongxia Wang, Jun Sun, and Peng Cheng. 2021. Robot: robustness-oriented testing for deep learning systems. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 300–311.
- [75] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. 2016. How to use t-SNE effectively. *Distill* 1, 10 (2016), e2.
- [76] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering*. Vol. 9783642290. 1–236 pages. <https://doi.org/10.1007/978-3-642-29044-2>
- [77] Bowen Xu, Fanghong Guo, Changyun Wen, and Wen-An Zhang. 2021. Stealthy False Data Injection Attack Detection in Smart Grids with Uncertainties: A Deep Transfer Learning Based Approach. *arXiv preprint arXiv:2104.06307* (2021).
- [78] Jian Yang and Jing-yu Yang. 2003. Why can LDA be performed in PCA transformed space? *Pattern recognition* 36, 2 (2003), 563–566.
- [79] Matthew D. Zeiler and Rob Fergus. 2014. Visualizing and Understanding Convolutional Networks. In *Computer Vision – ECCV 2014*, David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars (Eds.). Springer International Publishing, Cham, 818–833.
- [80] Hao Zhang and WK Chan. 2019. Apricot: a weight-adaptation approach to fixing deep learning models. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 376–387.
- [81] Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. 2020. Machine Learning Testing: Survey, Landscapes and Horizons. *IEEE Transactions on Software Engineering* (2020), 1–1. <https://doi.org/10.1109/TSE.2019.2962027>
- [82] Xiaoyu Zhang, Juan Zhai, Shiqing Ma, and Chao Shen. 2021. AUTOTRAINER: An Automatic DNN Training Problem Detection and Repair System. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 359–371.
- [83] Yu-Dong Zhang, Suresh Chandra Satapathy, David S Guttery, Juan Manuel Górriz, and Shui-Hua Wang. 2021. Improved breast cancer classification through combining graph convolutional network and convolutional neural network. *Information Processing & Management* 58, 2 (2021), 102439.
- [84] Yue Zhao, Hong Zhu, Kai Chen, and Shengzhi Zhang. 2021. AI-Lancet: Locating Error-inducing Neurons to Optimize Neural Networks. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 141–158.
- [85] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. 2016. Learning Deep Features for Discriminative Localization. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2921–2929. <https://doi.org/10.1109/CVPR.2016.319>
- [86] Tahereh Zohdinasab, Vincenzo Riccio, Alessio Gambi, and Paolo Tonella. 2021. Deephyperion: exploring the feature space of deep learning-based systems through illumination search. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 79–90.