Explaining Image Classifiers Using Statistical Fault Localization

Youcheng Sun[1(✉)] , Hana Chockler[2] , Xiaowei Huang[3] ,
and Daniel Kroening[4]

[1] Queen's University Belfast, Belfast, Northern Ireland
youcheng.sun@qub.ac.uk
[2] King's College London, London, England
[3] University of Liverpool, Liverpool, England
[4] University of Oxford, Oxford, England

Abstract. The black-box nature of deep neural networks (DNNs) makes it impossible to understand *why* a particular output is produced, creating demand for "Explainable AI". In this paper, we show that statistical fault localization (SFL) techniques from software engineering deliver high quality explanations of the outputs of DNNs, where we define an explanation as a minimal subset of features sufficient for making the same decision as for the original input. We present an algorithm and a tool called DEEPCOVER, which synthesizes a ranking of the features of the inputs using SFL and constructs explanations for the decisions of the DNN based on this ranking. We compare explanations produced by DEEPCOVER with those of the state-of-the-art tools GradCAM, LIME, SHAP, RISE and Extremal and show that explanations generated by DEEP-COVER are consistently better across a broad set of experiments. On a benchmark set with known ground truth, DEEPCOVER achieves 76.7% accuracy, which is 6% better than the second best Extremal.

Keywords: Deep learning · Explainability · Statistical fault localization · Software testing

1 Introduction

Deep neural networks (DNNs) are increasingly used in place of traditionally engineered software in many areas. DNNs are complex non-linear functions with algorithmically generated (and not engineered) coefficients, and therefore are effectively "black boxes". They are given an input and produce an output, but the calculation of these outputs is difficult to explain [26]. The goal of *explainable AI* is to create artifacts that provide a rationale for why a neural network generates a particular output for a particular input. This is argued to enable stakeholders to understand and appropriately trust neural networks.

A typical use-case of DNNs is classification of highly dimensional inputs, such as images. DNNs are multi-layered networks with a predefined structure

that consists of layers of neurons. The coefficients for the neurons are determined by a training process on a data set with given classification labels. The standard criterion for the adequacy of training is the accuracy of the network on a separate validation data set. This criterion is clearly only as comprehensive as the validation data set. In particular, this approach suffers from the risk that the validation data set is lacking an important instance [36]. Explanations provide additional insight into the decision process of a neural network [9,23].

In traditional software development, SFL measures have a substantial track record of helping engineers to debug sequential programs [19]. These measures rank program locations by counting the number of times a particular location is visited in passing and in failing executions for a given test suite and applying statistical formulae. The ranked list is presented to the engineer. The main advantage of SFL measures is that they are comparatively inexpensive to compute. There are more than a hundred of measures in the literature [33]. Some of the most widely used measures are Zoltar, Ochiai, Tarantula and Wong-II [8,14,21,34].

Our Contribution. We propose to apply the concept of *explanations* introduced by Halpern and Pearl in the context of *actual causality* [11]. Specifically, we define an explanation as a subset of features of the input that is *sufficient* (in terms of explaining the cause of the outcome), *minimal* (i.e., not containing irrelevant or redundant elements), and *not obvious.*

Using this definition and SFL measures, we have developed **DeepCover** – a tool that provides explanations for DNNs that classify images. DeepCover ranks the pixels using four well-known SFL measures (Zoltar, Ochiai, Tarantula and Wong-II) based on the results of running test suites constructed from random mutations of the input image. DeepCover then uses this ranking to efficiently construct an approximation of the explanation (as explained below, the exact computation is intractable).

We compare the quality of the explanations produced by DeepCover with those generated by the state-of-the-art tools GradCAM, LIME, SHAP, RISE and Extremal in several complementary scenarios. First, we measure the size of the explanations as an indication of the quality of the explanations. To complement this setup, we further apply the explanation tools to the problem of weakly supervised object localization (WSOL). We also create a "chimera" benchmark, consisting of images with a known ground truth. DeepCover exhibits consistently better performance in these evaluations. Finally, we investigate the use of explanations in a DNN security application, and show that DeepCover successfully identifies the backdoors that trigger Trojaning attacks.

2   Related Work

There is a large number of methods for explaining DNN decisions. Our approach belongs to a category of methods that compute local perturbations. Such methods compute and visualize the important features of an input instance to

explain the corresponding output. Given a particular input, LIME [27] samples the neighborhood of this input and creates a linear model to approximate the model's local behavior; owing to the high computational cost of this approach, the ranking uses super-pixels instead of individual pixels. In [4], the natural distribution of the input is replaced by a user-defined distribution and the Shapley Value method is used to analyze combinations of input features and to rank their importance. In [3], the importance of input features is estimated by measuring the flow of information between inputs and outputs. Both the Shapley Value and the information-theoretic approaches are computationally expensive. In RISE [25], the importance of a pixel is computed as the expectation over all local perturbations conditioned on the event that the pixel is observed. More recently, the concept of "extreme perturbations" has been introduced to improve the perturbation analysis by the EXTREMAL algorithm [6].

On the other hand, gradient-based methods only need one backward pass. GRADCAM [29] passes the class-specific gradient into the final convolutional layer of a DNN to coarsely highlight important regions of an input image. In [30], the activation of each neuron is compared with some reference point, and its contribution score for the final output is assigned according to the difference. The work of [4,18,27,30] is similar: an approximation of the model's local behavior using a simpler linear model and an application of the Shapley Value theory to solve this model.

Our algorithm for generating explanations is inspired by the statistical fault localization (SFL) techniques in software testing [19] (see Sect. 3.2 for an overview). SFL measures have the advantage of being simple and efficient. They are widely used for localizing causes of software failures. Moreover, there are single-bug optimal measures [15] that guarantee that the fault is localized when it is the single cause for the program failure. While it is not always possible to localize a single best feature to explain a DNN image classifier, single-bug optimal measures often perform well even when there is more than one fault in the program [16]. From the software engineering perspective, our work can be regarded as applying SFL techniques for diagnosing the neural network's decision. This complements recent works on the testing and validation of AI [20,22,31,32], for which a detailed survey can be found in [13].

3 Preliminaries

3.1 Deep Neural Networks (DNNs)

We briefly review the relevant definitions of deep neural networks. Let $f : \mathcal{I} \to \mathcal{O}$ be a deep neural network $\mathcal{N}$ with N layers. For a given input $x \in \mathcal{I}$, $f(x) \in \mathcal{O}$ calculates the output of the DNN, which could be, for instance, a classification label. Images are among the most popular inputs for DNNs, and in this paper we focus on DNNs that classify images. Specifically, we have

$$f(x) = f_N(\ldots f_2(f_1(x; W_1, b_1); W_2, b_2) \ldots; W_N, b_N) \tag{1}$$

where W_i and b_i for $i = 1, 2, \ldots, N$ are learnable parameters, and $f_i(z_{i-1}; W_{i-1}, b_{i-1})$ is the layer function that maps the output of layer $(i-1)$, i.e., z_{i-1}, to the input of layer i. The combination of the layer functions yields a highly complex behavior, and the analysis of the information flow within a DNN is challenging. There is a variety of layer functions for DNNs, including fully connected layers, convolutional layers and max-pooling layers. Our algorithm is independent of the specific internals of the DNN and treats a given DNN as a black box.

3.2 Statistical Fault Localization (SFL)

Statistical fault localization techniques (SFL) [19], have been widely used in software testing to aid in locating the causes of failures of programs. SFL techniques rank program elements (e.g., statements or assignments) based on their *suspiciousness scores*. Intuitively, a program element is more suspicious if it appears in failed executions more frequently than in correct executions (the exact formulas for ranking differ). Diagnosis of the faulty program can then be conducted by manually examining the ranked list of elements in descending order of their suspiciousness until the culprit for the fault is found.

The SFL procedure first executes the program under test using a set of inputs. It records the program executions together with a set of Boolean flags that indicate whether a particular element was executed by the current test. The task of a fault localization tool is to compute a ranking of the program elements based on the values of these flags. Following the notation in [19], the suspiciousness score of each program statement s is calculated from a set of parameters $\langle a_{ep}^s, a_{ef}^s, a_{np}^s, a_{nf}^s \rangle$ that give the number of times the statement s is executed (e) or not executed (n) on passing (p) and on failing (f) tests. For instance, a_{ep}^s is the number of tests that passed and executed s.

A large number of measures have been proposed to calculate the suspiciousness scores. In Eq. 2 we list the most widely used ones [8,14,21,34]; those are also the measures that we use in our ranking procedure.

$$\text{Ochiai: } \frac{a_{ef}^s}{\sqrt{(a_{ef}^s + a_{nf}^s)(a_{ef}^s + a_{ep}^s)}} \tag{2a}$$

$$\text{Tarantula: } \frac{\frac{a_{ef}^s}{a_{ef}^s + a_{nf}^s}}{\frac{a_{ef}^s}{a_{ef}^s + a_{nf}^s} + \frac{a_{ep}^s}{a_{ep}^s + a_{np}^s}} \tag{2b}$$

$$\text{Zoltar: } \frac{a_{ef}^s}{a_{ef}^s + a_{nf}^s + a_{ep}^s + \frac{10000 a_{nf}^s a_{ep}^s}{a_{ef}^s}} \tag{2c}$$

$$\text{Wong-II: } a_{ef}^s - a_{ep}^s \tag{2d}$$

There is no single best measure for fault localization. Different measures perform better on different applications, and best practice is to use them together.

4   What Is an Explanation?

An explanation of an output of an automated procedure is essential in many areas, including verification, planning, diagnosis and the like. A good explanation can increase a user's confidence in the result. Explanations are also useful for determining whether there is a fault in the automated procedure: if the explanation does not make sense, it may indicate that the procedure is faulty. It is less clear how to define what a *good* explanation is. There have been a number of definitions of explanations over the years in various domains of computer science [2,7,24], philosophy [12] and statistics [28]. The recent increase in the number of machine learning applications and the advances in deep learning led to the need for *explainable AI*, which is advocated, among others, by DARPA [9] to promote understanding, trust, and adoption of future autonomous systems based on learning algorithms (and, in particular, image classification DNNs). DARPA provides a list of questions that a good explanation should answer and an epistemic state of the user after receiving a good explanation. The description of this epistemic state boils down to *adding useful information* about the output of the algorithm and *increasing trust* of the user in the algorithm.

In this paper, we are loosely adopting the definition of explanations by Halpern and Pearl [11], which is based on their definition of actual causality [10]. Roughly speaking, they state that a good explanation gives an answer to the question *"why did this outcome occur"*, which is similar in spirit to DARPA's informal description. As we do not define our setting in terms of actual causality, we omit the parts of the definition that refer to causal models and causal settings. The remaining parts of the definition of explanation are:

1. an explanation is a *sufficient* cause of the outcome;
2. an explanation is a *minimal* such cause (that is, it does not contain irrelevant or redundant elements);
3. an explanation is *not obvious*; in other words, before being given the explanation, the user could conceivably imagine other explanations for the outcome.

In image classification using DNNs, the non-obviousness holds for all but extremely trivial images. Translating 1) and 2) into our setting, we get the following definition.

Definition 1. *An explanation in image classification is a minimal subset of pixels of a given input image that is sufficient for the DNN to classify the image, where "sufficient" is defined as containing only this subset of pixels from the original image, with the other pixels set to the background colour.*

We note that (1) the explanation cannot be too small (or empty), as a too small subset of pixels would violate the sufficiency requirement, and (2) there can be multiple explanations for a given input image.

The precise computation of an explanation in our setting is intractable, as it is equivalent to the earlier definition of explanations in binary causal models, which is DP-complete [5]. A brute-force approach of checking all subsets of pixels

of the input image is exponential in the size of the image. In Sect. 5 we describe an efficient linear-time approach to computing an approximation of an explanation and argue that for practical purposes, this approximation is sufficiently close to an exact explanation as defined above.

5   SFL Explanation for DNNs

We propose a *black-box explanation technique* based on statistical fault localization. In traditional software development, SFL measures are used for ranking program elements that cause a failure. In our setup, the goal is different: we are searching for an explanation of why a particular input to a given DNN yields a particular output; our technique is agnostic to whether the output is correct. We start with describing our algorithm on a high level and then present the pseudo-code and technical details.

Generating the Test Suite. SFL requires test inputs. Given an input image x that is classified by the DNN $\mathcal{N}$ as $y = \mathcal{N}[x]$, we generate a set of images by *randomly mutating* x. A *legal mutation* masks a subset of the pixels of x, i.e., sets these pixels to the background color. The DNN computes an output for each mutant; we annotate it with "y" if that output matches that of x, and with "$\neg y$" to indicate that the output differs. The resulting test suite $T(x)$ of annotated mutants is an input to the DEEPCOVER algorithm.

Ranking the Pixels of x. We assume that the original input x consists of n pixels $\mathcal{P} = \{p_1, \ldots, p_n\}$. Each test input $t \in T(x)$ exhibits a particular spectrum for the pixel set, in which some pixels are the same as in the original input x and others are masked. The presence or masking of a pixel in x may affect the output of the DNN.

We use SFL measures to rank the set of pixels of x by slightly abusing the notions of passing and failing tests. For a pixel p_i of x, we compute the vector $\langle a_{ep}^i, a_{ef}^i, a_{np}^i, a_{nf}^i \rangle$ as follows:

- a_{ep}^i is the number of mutants in $T(x)$ labeled y in which p_i is not masked;
- a_{ef}^i is the number of mutants in $T(x)$ labeled $\neg y$ in which p_i is not masked;
- a_{np}^i is the number of mutants in $T(x)$ labeled y in which p_i is masked;
- a_{nf}^i is the number of mutants in $T(x)$ labeled $\neg y$ in which p_i is masked.

Once we construct the vector $\langle a_{ep}^i, a_{ef}^i, a_{np}^i, a_{nf}^i \rangle$ for every pixel, we apply the SFL measures discussed in Sect. 3.2 to rank the pixels of x for their importance regarding the DNN's output (the importance corresponds to the suspiciousness score computed by SFL measures).

Constructing an Explanation. An explanation is constructed by iteratively adding pixels to the set in the descending order of their ranking (that is, we start with the highest-ranked pixels) until the set becomes sufficient for the DNN to classify the image. This set is presented to the user as an explanation.

Algorithm 1. SFL Explanation for DNNs

INPUT: DNN $\mathcal{N}$, image x, SFL measure M
OUTPUT: a subset of pixels $\mathcal{P}^{exp}$

```
 1: T(x) ← test_inputs_gen(N, x)
 2: for each pixel pᵢ ∈ P do
 3:     calculate aⁱₑₚ, aⁱₑf, aⁱₙₚ, aⁱₙf from T(x)
 4:     valueᵢ ← M(aⁱₑₚ, aⁱₑf, aⁱₙₚ, aⁱₙf)
 5: end for
 6: pixel_ranking ← pixels in P from high value to low
 7: Pᵉˣᵖ ← ∅
 8: for each pixel pᵢ ∈ pixel_ranking do
 9:     Pᵉˣᵖ ← Pᵉˣᵖ ∪ {pᵢ}
10:     xᵉˣᵖ ← mask pixels of x that are not in Pᵉˣᵖ
11:     if N[xᵉˣᵖ] = N[x] then
12:         return Pᵉˣᵖ
13:     end if
14: end for
```

5.1 SFL Explanation Algorithm

We now present our algorithms in detail. Algorithm 1 starts by calling procedure *test_inputs_gen* to generate the set $T(x)$ of test inputs (Line 1). It then computes the vector $\langle a^i_{ep}, a^i_{ef}, a^i_{np}, a^i_{nf} \rangle$ for each pixel $p_i \in \mathcal{P}$ using $T(x)$ (Lines 2–5). Next, the algorithm computes the ranking of each pixel according to the specified measure M (Line 6). Formulas for measures are as in Eq. (2a)–(2d). The pixels are sorted in descending order of their ranking (from high *value* to low *value*).

From Line 7 onward in Algorithm 1, we construct a subset of pixels $\mathcal{P}^{exp}$ to explain $\mathcal{N}$'s output on this particular input x as follows. We add pixels to $\mathcal{P}^{exp}$, while $\mathcal{N}$'s output on $\mathcal{P}^{exp}$ does not match $\mathcal{N}[x]$. This process terminates when $\mathcal{N}$'s output is the same as on the whole image x. Finally, $\mathcal{P}^{exp}$ is returned as the explanation. At the end of this section we discuss why $\mathcal{P}^{exp}$ is not a precise explanation according to Definition 1 and argue that it is a good approximation (coinciding with a precise explanation in most cases).

As the quality of the ranked list computed by SFL measures inherently depends on the quality of the test suite, the choice of the set $T(x)$ of mutant images plays an important role in our SFL explanation algorithm for DNNs. While it is beyond the scope of this paper to identify the best set $T(x)$, we propose an effective method for generating $T(x)$ in Algorithm 2. The core idea of Algorithm 2 is to balance the number of test inputs annotated with "y" (that play the role of the passing traces) with the number of test inputs annotated with "$\neg y$" (that play the role of the failing traces). Its motivation is that, when applying fault localisation in software debugging, the rule of thumb is to maintain a balance between passing and failing cases.

The fraction σ of the set of pixels of x that are going to be masked in a mutant is initialized by a random or selected number between 0 and 1 (Line 2)

Algorithm 2. *test_inputs_gen*$(\mathcal{N}, x)$

INPUT: DNN $\mathcal{N}$, image x (with n pixels)
OUTPUT: test suite $T(x)$
PARAMETERS: σ, ϵ, test suite size m

```
 1: T(x) ← ∅
 2: σ ← sample in the range (0, 1)
 3: while |T(x)| < m do
 4:     x′ ← randomly select and mask σ · n pixels in x
 5:     T(x) ← T(x) ∪ {x′}
 6:     if 𝒩[x′] ≠ 𝒩[x] then
 7:         σ ← max{σ − ϵ, 0}
 8:     else
 9:         σ ← min{σ + ϵ, 1}
10:     end if
11: end while
12: return T(x)
```

---

and is later updated at each iteration according to the decision of $\mathcal{N}$ on the previously constructed mutant. In each iteration of the algorithm, a randomly chosen set of $(\sigma \cdot n)$ pixels in x is masked and the resulting new input x' is added to $T(x)$ (Lines 4–5). Roughly speaking, if a mutant is not classified with the same label as x, we decrease the fraction of masked pixels by a pre-defined small number ϵ; if the mutant is classified with the same label as x, we increase the fraction of masked pixels by the same ϵ.

5.2   Relationship Between $\mathcal{P}^{exp}$ and Definition 1

Recall that Definition 1 requires an explanation to be *sufficient*, *minimal*, and *not obvious* (see Sect. 4). As we argued above, the non-obviousness requirement holds for all but very simple images. It is also easy to see that $\mathcal{P}^{exp}$ is sufficient, since this is a stopping condition for adding pixels to this set (Line 11 in Algorithm 1).

The only condition that might not hold is minimality. The reason for possible non-minimality is that the pixels of x are added to the explanation in the order of their ranking, with the highest-ranking pixels being added first. It is therefore possible that there is a high-ranked pixel that was added in one of the previous iterations, but is now not necessary for the correct classification of the image (note that the process of adding pixels to the explanation stops when the DNN successfully classifies the image; this, however, shows minimality only with respect to the order of addition of pixels). We believe that the redundancy resulting from our approach is likely to be small, as higher-ranked pixels have a larger effect on the DNN's decision. In fact, even if our explanation is, strictly speaking, not minimal, it might not be a disadvantage, as it was found that humans prefer explanations with some redundancy [35].

Another advantage of our algorithm is that its running time is linear in the size of the set $T(x)$ and the size of the image, hence it is much more efficient than the brute-force computation of all explanations as described in Sect. 4 (and in fact, any algorithm that computes a precise explanation, as the problem is intractable). One hypothetical advantage of the enumeration algorithm is that it can produce all explanations; however, multiple explanations do not necessarily provide better insight into the decision process.

6   Experimental Evaluation

We have implemented the SFL explanation algorithm for DNNs presented in Sect. 5 in the tool DEEPCOVER[1]. We now present the experimental results. We tested DEEPCOVER on a variety of DNN models for ImageNet and we compare DEEPCOVER with the most popular and most recent work in AI explanation: LIME [27], SHAP [18], GradCAM [29], RISE [25] and Extremal [6].[2]

6.1   Experimental Setup

We configure the heuristic test generation in Algorithm 2 with $\sigma = \frac{1}{5}$ and $\epsilon = \frac{1}{6}$, and the size m of the test set $T(x)$ is 2,000. These values have been chosen empirically and remain the same through all experiments. It is possible that they are not appropriate for all input images, and that for some inputs increasing m or tuning σ and ϵ produces a better explanation. All experiments are run on a laptop with a 3.9 GHz Intel i7-7820HQ and 16 GB of memory.

6.2   Are the Explanations from DeepCover Useful?

Figure 1 showcases representative output from DEEPCOVER on the Xception model. We can say that explanations are indeed useful and meaningful. Each subfigure in Fig. 1 provides the original input and the output of DEEPCOVER. We highlight misclassifications and counter-intuitive explanations in red. One of the more interesting examples is the "cowboy hat"image. Although Xception labels the input image correctly, an explanation produced by DEEPCOVER indicates that this decision is not based on the correct feature (the hat in the image), but on the face, which is an unexpected feature for the label 'cowboy hat'. While this image was not, technically speaking, misclassified, the explanation points to a flaw in the DNN's reasoning. The "wool" and "whistle" are two misclassifications by Xception, and the explanations generated by DEEPCOVER can help us to understand why the misclassification happens: there are similarities between the features that are used for the correct and the incorrect labels.

---

[1] https://github.com/theyoucheng/deepcover.

[2] LIME version 0.1.33; SHAP version 0.29.1; GradCAM, RISE and Extremal are from https://github.com/facebookresearch/TorchRay (commit 6a198ee61d229360a3def59 0410378d2ed6f1f06).
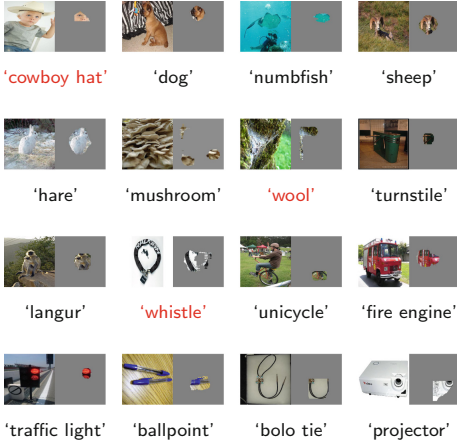
Fig. 1. Input images and explanations from DeepCover for Xception (red labels highlight misclassification or counter-intuitive explanations) (Color figure online)
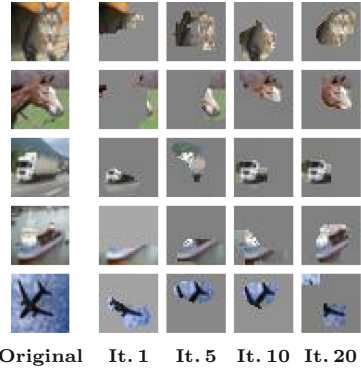

Fig. 2. Explanations of the DNN at different training stages: the 1st column are the original images and the subsequent columns give the explanations for a particular training iteration (CIFAR-10 validation data set)

Furthermore, we apply DeepCover after each training iteration to the intermediate DNN. In Fig. 2 we showcase some representative results at different stages of the training. Overall, as the training procedure progresses, explanations of the DNN's decisions focus more on the "meaningful" part of the input image, e.g., those pixels contributing to the interpretation of image (see, for example, the progress of the training reflected in the explanations of DNN's classification of the first image as a 'cat'). This result reflects that the DNN is being trained to learn features of different classes of inputs. Interestingly, we also observed that the DNN's feature learning is not always monotonic, as demonstrated in the bottom row of Fig. 2: after the 10th iteration, explanations for the DNN's classification of an input image as an 'airplane' drift away from the intuitive parts of the input towards pixels that may not fit human interpretation (we repeated the experiments multiple times to minimize the uncertainty because of the randomization in our SFL algorithm). The explanations generated by DeepCover may thus be useful for assessing the adequacy of the DNN training: they allow us to check, whether the DNN is aligned with the developer's intent during training. Additionally, the results in Fig. 2 satisfy the "sanity" requirement postulated in [1]: the explanations from DeepCover evolve when the model parameters change during the training.

6.3 Comparison with the State-of-the-art

We compare DeepCover with state-of-the-art DNN explanation tools. The DNN is VGG16 and we randomly sample 1,000 images from ILSVRC2012 as

inputs. We evaluate the effect of highly ranked features by different methods following an addition/deletion style experiment [6,25].

An explanation computed by Algorithm 1 is a subset $\mathcal{P}^{exp}$ of top-ranked pixels out of the set $\mathcal{P}$ of all 224×224 pixels that is sufficient for the DNN to classify the image correctly. We define the size of the explanation as $\frac{|\mathcal{P}^{exp}|}{|\mathcal{P}|}$. We use the size of an explanation as a proxy for its quality.
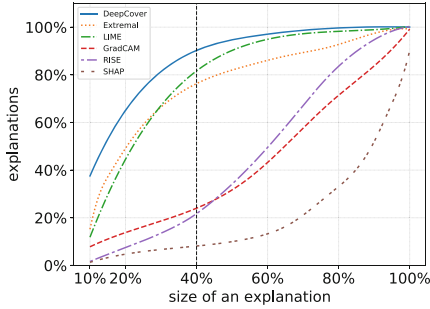

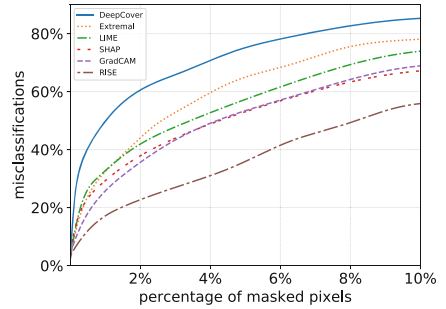**Fig. 3.** Comparison in the size of generated explanations by different tools

Fig. 4. Misclassification vs percentage of masked pixels for different tools

Figure 3 compares DeepCover and its competitors with respect to the size of the generated explanations. The position on the x-axis is the size of the explanation, and the position on the y-axis gives the accumulated percentage of explanations: that is, all generated explanations with smaller or equal size.

The data in Fig. 3 suggests that explanations based on SFL ranking are superior in terms of their size. For example, nearly 40% of the DNN inputs can be explained via DeepCover using no more than 10% of the total input pixels, which is two times as good as the second best explanation method Extremal.

We quantify the degree of redundancy in the generated explanations as follows. We mask pixels following the ranking generated by the different methods until we obtain a different classification. The smaller the number of pixels that have to be masked, the more important the highest-ranked features are. We present the number of pixels changed (normalized over the total number of pixels) in Fig. 4. Again, DeepCover dominates the others. Using DeepCover's ranking, the classification is changed after masking no more than 2% of the total pixels in 60% of the images. To achieve the same classification outcomes, the second best method Extremal requires changing 4% of the total number of pixels, and that is twice the number of pixels needed by DeepCover.

Discussion. We have refrained from using human judges to assess the quality of the explanations, and instead use size as a proxy measure to quantify the quality of explanation. However, a smaller explanation is not necessarily a better explanation—in fact, "people have a preference for explanations with some

redundancy" [35]. We therefore complement our evaluation with further experiments. Figure 5 gives the results of using the explanations for the weakly supervised object localization (WSOL). We measure the intersection of union (IoU) between the object bounding box and the equivalent number of top-ranked pixels. The IoU is a standard measure of success in object detection and a higher IoU is better. The results confirm again that the top-ranked pixels from DEEPCOVER perform better than those generated by other tools.

Comparison with Rise. The RISE tool generates random masks and calculates a ranking of the input pixels using the expected confidence of the classification of the masked images. A rank of a pixel p by RISE depends only on the confidence of the images in which p is unmasked. By contrast, DEEPCOVER uses a binary classification (a mutant image is either classified the same as the original image or not) and takes into account both the images where p is masked and where it is unmasked. Figures 3 and 4 demonstrate that DEEPCOVER outperforms RISE, producing smaller and more intuitive explanations. Furthermore, the DEEPCOVER approach is more general and does not depend on a particular sampling distribution as long as its mutant test suite is balanced (Sect. 5.1). Moreover, the DEEPCOVER approach is less sensitive to the size of the mutant test suite (Fig. 6). When the size of the test suite decreased from 2,000 to 200, the size of the generated explanation only increased by 3% of the total pixels on average.
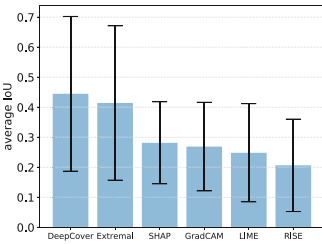

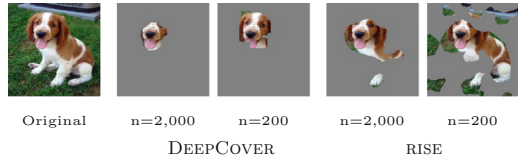**Fig. 5.** Results for weakly supervised object localisation

Fig. 6. Explanations for the 'Welsh springer spaniel' by DEEPCOVER and RISE with varying number of samples (i.e. n)

Next, we present a synthetic benchmark (Sect. 6.4) and a security application (Sect. 6.5).

6.4   Generating "ground Truth" with a Chimera Benchmark

The biggest challenge in evaluating explanations for DNNs (and even for human decision making) is the lack of the *ground truth*. Human evaluations of the explanations remain the most widely accepted measure, but are often subjective. In
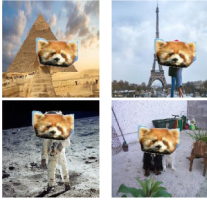
Fig. 7. Examples of embedding the red panda (Color figure online)

Table 1. IoUs between the embedded red panda and the highest ranked pixels for four different tools

	IoU ≥ 0.5	IoU ≥ 0.6	IoU ≥ 0.7
DEEPCOVER	**76.7%**	**54.9%**	9.8%
Extremal	70.7%	21.5%	2.2%
RISE	55.8%	42.9%	**25.7%**
GradCAM	0%	0%	0%

the experiment we describe below, we synthesize a *Chimera benchmark*[3] by randomly superimposing a "red panda" explanation (a part of the image of the red panda) onto a set of randomly chosen images. The benchmark consists of $1,000$ composed (aka "Chimera") images that retain the "red panda" label when using both the MobileNet and the VGG16 classifiers. Figure 7 gives several examples of the Chimera images. The rationale is that if such an image is indeed classified as "red panda" by the DNN, then the explanation of this classification must be contained among the pixels we have superimposed onto the original image.

For each image from the Chimera benchmark, we rank its pixels using DEEP-COVER and other tools. We then check whether any of their top-π highest ranked pixels are part of the "red panda". In Table 1, we measure the IoU (intersection of union) between the ground truth explanation and the top-π highest ranked pixels, where π ranges from 1% to 100%. Assuming that an IoU ≥ 0.5 is a successful detection, DEEPCOVER successfully detects the ground truth planted in the image in 76.7% of the total cases and it is 6% better than the second best Extremal. The benefit provided by DEEPCOVER is even more substantial when requiring 0.6 IoU. Overall, the results in Table 1 are consistent with the addition/deletion experiment (Figs. 3 and 4) and the WSOL experiment (Fig. 5), with DEEPCOVER topping the list. Interestingly, when RISE succeeds to find the explanation, it seems to localize it better (with IoU ≥ 0.7). GradCAM fails to detect the embedded red panda in all cases. These observations support the hypothesis that a benchmark like Chimera is a good approximation for ground truth, and helps us to compare algorithmic alternatives.

6.5 Trojaning Attacks

The authors of [17] say that a DNN is "trojaned" if it behaves correctly on ordinary input images but exhibits malicious behavior when a "Trojan trigger" is part of the input. Thus, we can treat this trigger as a ground truth explanation for the Trojaned behavior of the DNN. We have applied DEEPCOVER to identify the embedded trigger in the input image for the Trojaned VGG Face [17]. The result is illustrated in Fig. 8. This use case suggests that there is scope for the application of DEEPCOVER in DNN security.

---

[3] The benchmark images are publicly available at http://www.roaming-panda.com/.

(a)   (b)

Fig. 8. Applying DEEPCOVER to Trojaning attacks on VGG Face. The Trojan trigger is the square shape in the lower right corner of the image; the DEEPCOVER explanation for the Trojan behaviour is on the right.

When applying DEEPCOVER to the Trojaned data set in [17], the top 8% highest ranked pixels have an average IoU value of 0.6 with the Trojan trigger. According to DEEPCOVER, the Trojaning output for each input is caused by a small part of its embedded trigger. This black-box discovery by DEEPCOVER is consistent with and further optimizes the theory of DNN Trojaning [17]. Finally, as many as 80% of the (ground truth) Trojan triggers are successively localized (with IoU $\geq$ 0.5) by only $\pi = 8\%$ of the pixels top-ranked by DEEPCOVER. DEEPCOVER is thus very effective.

6.6   Threats to Validity

In this part, we highlight several threats to the validity of our evaluation.
Lack of Ground Truth. We have no ground truth for evaluating the generated explanations for Xception on ImageNet images, hence we use the size of an explanation as a proxy. We have the ground truth for the Chimera images of red panda (Fig. 7) and for the Trojaning attacks (Fig. 8), and the results support our claims of the high quality of DEEPCOVER explanations.

Selection of SFL Measures. We have only evaluated four SFL measures (Ochiai, Zoltar, Tarantula and Wong-II). There are hundreds more such measures, which may reveal new observations.

Selection of Parameters When Generating Test Inputs. When generating the test suite $T(x)$, we empirically configure the parameters in the test generation algorithm. The choice of parameters affects the results of the evaluation and they may be overfitted.

7   Conclusions

This paper advocates the application of statistical fault localization (SFL) for the generation of explanations of the output of neural networks. Our definition of explanations is inspired by actual causality, and we demonstrate that we can efficiently compute a good approximation of a precise explanation using a lightweight ranking of features of the input image based on SFL measures. The algorithm is implemented in the tool DEEPCOVER. Extensive experimental

results demonstrate that DEEPCOVER consistently outperforms other explanation tools and that its explanations are accurate when compared to ground truth (that is, the explanations of the images have a large overlap with the explanation planted in the image).

References

1. Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., Kim, B.: Sanity checks for saliency maps. In: Advances in Neural Information Processing Systems, pp. 9505–9515 (2018)
2. Chajewska, U., Halpern, J.Y.: Defining explanation in probabilistic systems. In: Uncertainty in Artificial Intelligence (UAI), pp. 62–71. Morgan Kaufmann (1997)
3. Chen, J., Song, L., Wainwright, M., Jordan, M.: Learning to explain: an information-theoretic perspective on model interpretation. In: International Conference on Machine Learning (ICML), vol. 80, pp. 882–891. PMLR (2018)
4. Datta, A., Sen, S., Zick, Y.: Algorithmic transparency via quantitative input influence: theory and experiments with learning systems. In: Security and Privacy (S&P), pp. 598–617. IEEE (2016)
5. Eiter, T., Lukasiewicz, T.: Complexity results for explanations in the structural-model approach. Artif. Intell. **154**(1–2), 145–198 (2004)
6. Fong, R., Patrick, M., Vedaldi, A.: Understanding deep networks via extremal perturbations and smooth masks. In: International Conference on Computer Vision (ICCV), pp. 2950–2958. IEEE (2019)
7. Gärdenfors, P.: Knowledge in Flux. MIT Press, Cambridge (1988)
8. Gonzalez-Sanchez, A.: Automatic error detection techniques based on dynamic invariants. M.S. thesis, Delft University of Technology, The Netherlands (2007)
9. Gunning, D.: Explainable artificial intelligence (XAI) - program information. Defense Advanced Research Projects Agency (2017). https://www.darpa.mil/program/explainable-artificial-intelligence
10. Halpern, J.Y., Pearl, J.: Causes and explanations: a structural-model approach. Part I: causes. Br. J. Philos. Sci. **56**(4), 843–887 (2005)
11. Halpern, J.Y., Pearl, J.: Causes and explanations: a structural-model approach. Part II: explanations. Br. J. Philos. Sci. **56**(4), 889–911 (2005)
12. Hempel, C.G.: Aspects of Scientific Explanation. Free Press, New York (1965)
13. Huang, X., et al.: A survey of safety and trustworthiness of deep neural networks: verification, testing, adversarial attack and defence, and interpretability. Comput. Sci. Rev. **37**, 100270 (2020)
14. Jones, J.A., Harrold, M.J.: Empirical evaluation of the Tarantula automatic fault-localization technique. In: Proceedings of ASE, pp. 273–282. ACM (2005)
15. Landsberg, D., Chockler, H., Kroening, D., Lewis, M.: Evaluation of measures for statistical fault localisation and an optimising scheme. In: Egyed, A., Schaefer, I. (eds.) FASE 2015. LNCS, vol. 9033, pp. 115–129. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46675-9_8
16. Landsberg, D., Sun, Y., Kroening, D.: Optimising spectrum based fault localisation for single fault programs using specifications. In: Russo, A., Schürr, A. (eds.) FASE 2018. LNCS, vol. 10802, pp. 246–263. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-89363-1_14
17. Liu, Y., et al.: Trojaning attack on neural networks. In: Network and Distributed System Security Symposium (NDSS). The Internet Society (2018)

18. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Advances in Neural Information Processing Systems, pp. 4765–4774 (2017)

19. Naish, L., Lee, H.J., Ramamohanarao, K.: A model for spectra-based software diagnosis. ACM TOSEM **20**(3), 11 (2011)

20. Noller, Y., Păsăreanu, C.S., Böhme, M., Sun, Y., Nguyen, H.L., Grunske, L.: HyDiff: hybrid differential software analysis. In: Proceedings of the International Conference on Software Engineering (ICSE) (2020)

21. Ochiai, A.: Zoogeographic studies on the soleoid fishes found in Japan and its neighbouring regions. Bull. Jpn. Soc. Sci. Fish. **22**, 526–530 (1957)

22. Odena, A., Olsson, C., Andersen, D., Goodfellow, I.: TensorFuzz: debugging neural networks with coverage-guided fuzzing. In: International Conference on Machine Learning, pp. 4901–4911 (2019)

23. Olah, C., et al.: The building blocks of interpretability. Distill **3**, e10 (2018)

24. Pearl, J.: Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann, Burlington (1988)

25. Petsiuk, V., Das, A., Saenko, K.: RISE: randomized input sampling for explanation of black-box models. In: British Machine Vision Conference (BMVC). BMVA Press (2018)

26. Rahwan, I., et al.: Machine behaviour. Nature **568**(7753), 477 (2019)

27. Ribeiro, M.T., Singh, S., Guestrin, C.: Why should I trust you? Explaining the predictions of any classifier. In: Knowledge Discovery and Data Mining (KDD), pp. 1135–1144. ACM (2016)

28. Salmon, W.C.: Four Decades of Scientific Explanation. University of Minnesota Press, Minneapolis (1989)

29. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-CAM: visual explanations from deep networks via gradient-based localization. In: International Conference on Computer Vision (ICCV), pp. 618–626. IEEE (2017)

30. Shrikumar, A., Greenside, P., Kundaje, A.: Learning important features through propagating activation differences. In: International Conference on Machine Learning (ICML), vol. 70, pp. 3145–3153. PMLR (2017)

31. Sun, Y., Wu, M., Ruan, W., Huang, X., Kwiatkowska, M., Kroening, D.: Concolic testing for deep neural networks. In: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE, pp. 109–119 (2018)

32. Sun, Y., Zhou, Y., Maskell, S., Sharp, J., Huang, X.: Reliability validation of learning enabled vehicle tracking. In: International Conference on Robotics and Automation (ICRA). IEEE (2020)

33. Wong, W.E., Gao, R., Li, Y., Abreu, R., Wotawa, F.: A survey on software fault localization. IEEE TSE **42**(8), 707–740 (2016)

34. Wong, W.E., Qi, Y., Zhao, L., Cai, K.: Effective fault localization using code coverage. In: Computer Software and Applications Conference (COMPSAC), pp. 449–456 (2007)

35. Zemla, J.C., Sloman, S., Bechlivanidis, C., Lagnado, D.A.: Evaluating everyday explanations. Psychon. Bull. Rev. **24**(5), 1488–1500 (2017). https://doi.org/10.3758/s13423-017-1258-z

36. Ziegler, C.: A Google self-driving car caused a crash for the first time. The Verge (2016). https://www.theverge.com/2016/2/29/11134344/google-self-driving-car-crash-report