Explaining Deep Neural Networks Using Spectrum-Based Fault Localization

Youcheng Sun*, Hana Chockler†, Xiaowei Huang‡, Daniel Kroening§
*Queen's University Belfast, UK
youcheng.sun@qub.ac.uk
†King's College London, UK
hana.chockler@kcl.ac.uk
‡University of Liverpool, UK
xiaowei.huang@liverpool.ac.uk
§University of Oxford, UK
kroening@cs.ox.ac.uk

Abstract—**Deep neural networks (DNNs) increasingly replace traditionally developed software in a broad range of applications. However, in stark contrast to traditional software, the black-box nature of DNNs makes it impossible to understand their outputs, creating demand for "Explainable AI". Explanations of the outputs of the DNN are essential for the training process and are supporting evidence of the adequacy of the DNN. In this paper, we show that spectrum-based fault localization delivers good explanations of the outputs of DNNs. We present an algorithm and a tool PROTOZOA, which synthesizes a ranking of the parts of the inputs using several spectrum-based fault localization measures. We show that the highest-ranked parts provide explanations that are consistent with the standard definitions of explanations in the literature. Our experimental results on ImageNet show that the explanations we generate are useful visual indicators for the progress of the training of the DNN. We compare the results of PROTOZOA with SHAP and show that the explanations generated by PROTOZOA are on par or superior. We also generate adversarial examples using our explanations; the efficiency of this process can serve as a proxy metric for the quality of the explanations. Our measurements show that PROTOZOA's explanations yield a higher number of adversarial examples than those produced by SHAP.**

I. 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 output, but the functional processes that generate these outputs are difficult to explain [1]. 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 [2].

A typical use-case of DNNs is to classify 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 [3]. 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 [4], [5].

Explanations have been claimed to address this problem by providing additional insight into the decision process of a neural network [6], [7]. Explanations can be used to guide the training process to the missing inputs and to signal when the decisions are sufficiently accurate.

State of the art: The state of the art in producing explanations for image classifiers is an approach called *SHapley Additive exPlanations (SHAP)* [8], which assigns an "importance value" to each pixel. The algorithm treats the classification of a multi-dimensional input as a multi-player collaborative game, where each player represents a dimension. The importance value of a pixel is the contribution it makes to the classification. This method provides a reasonable, and accurate, explanation based on game theory. In practice, owing to the computational complexity of the algorithm, the implementation approximates the solution, leading to inaccuracies. In addition, since SHAP combines multiple techniques that are conceptually different, including [9]–[11], it is intrinsically complex.

In traditional software development, statistical fault localization measures have a substantial track record of finding causes of errors to aid in the debugging process of sequential programs [12]–[15]. 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 by applying a statistical formula. Hence, they are comparatively inexpensive to compute. Their precision depends on the quality of the test suite [16]–[21]. There are more than a hundred of measures mentioned in the literature [22]. Some of the most widely used measures are Zoltar, Ochiai, Tarantula, and Wong-II [23]–[26], which have been shown to outperform other measures in experimental comparisons.

Our contribution: We present **PROTOZOA**, which provides explanations for DNNs that classify images. Our explanations are synthesized from the ranking of image pixels by statistical fault localization measures using test suites constructed by randomly mutating a given input image. Our tool integrates four well-known measures (Zoltar, Ochiai, Tarantula and Wong-II). Experimental results on the ImageNet data set show that our explanations are visually better than those generated by SHAP. As "visually better" is not an objective metric, we measure the efficiency of the generation of adversarial example as a proxy for the quality of our explanations. While clearly not identical to "visually better", this metric has the advantage that it is objective and algorithmically computable. Our experimental results show that the explanations produced by PROTOZOA yield more adversarial examples than those produced by SHAP. An additional advantage of PROTOZOA is that it treats the DNN as a black-box, and that it is highly scalable.

The tool and the data for the experiments described in this paper, together with the scripts and the experimental setup, can be downloaded online[1].

II. Preliminaries

A. 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 still the most popular inputs for DNNs, and in this paper we focus on DNNs that classify images. Specifically, we have

$$f(x) = f_N(\dots f_2(f_1(x; W_1, b_1); W_2, b_2) \dots ; W_N, b_N) \quad (1)$$

where W_i and b_i for $i = 1, 2, \dots, N$ are learn-able 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 highly complex behavior, and the analysis of the information flow within a DNN is challenging. There are a variety of layer functions for DNNs, including, e.g., fully connected layers, convolutional layers and max-pooling layers.

Our algorithm is independent of the specific internals of the DNN. Given a particular input image x and $\mathcal{N}$'s output y, we present to the user a subset of the pixels of x that explain why $\mathcal{N}$ outputs y when given x. In the following, we use $\mathcal{N}[x]$ to denote the output of $\mathcal{N}$ for an input image x.

B. Spectrum-based fault localization (SBFL)

Our work is inspired by spectrum based fault localization [12]–[27], which has been widely used as an efficient approach to automatically locate root causes of failures of programs. SBFL 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 between the measures). Diagnosis of the faulty program can then be conducted by examining the ranked list of elements in descending order of their suspiciousness until the culprit for the fault is found.

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

The SBFL procedure first executes the program under test using a set of inputs. It records the program executions as *program spectra*, meaning that the execution is instrumented to modify a set of Boolean flags that indicate whether a particular program element was executed. The task of a fault localization tool is to compute a ranking of the program elements based on the program spectra. Following the notation in [12], the suspiciousness score of each program statement s is calculated from a set of parameters $\langle a^s_{ep}, a^s_{ef}, a^s_{np}, a^s_{nf} \rangle$ 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^s_{ep} is the number of tests that have passed and that have executed s.

A large number of measures has been proposed to calculate the suspicious score of each program element. We list below some of the most widely used measures; those are also the measures that we use in our ranking procedure.

$$\text{Ochiai [24]:} \quad \frac{a^s_{ef}}{\sqrt{(a^s_{ef} + a^s_{nf})(a^s_{ef} + a^s_{ep})}} \quad (2a)$$

$$\text{Zoltar [23]:} \quad \frac{a^s_{ef}}{a^s_{ef} + a^s_{nf} + a^s_{ep} + \frac{10000 a^s_{ef} a^s_{ep}}{a^s_{ef}}} \quad (2b)$$

$$\text{Tarantula [25]:} \quad \frac{\frac{a^s_{ef}}{a^s_{ef} + a^s_{nf}}}{\frac{a^s_{ef}}{a^s_{ef} + a^s_{nf}} + \frac{a^s_{ep}}{a^s_{ep} + a^s_{np}}} \quad (2c)$$

$$\text{Wong-II [26]:} \quad a^s_{ef} - a^s_{ep} \quad (2d)$$

The tool then presents to the user the list of program elements in the order of descending suspiciousness score. As reported in surveys [14], [15], [22], there is no single best measure for fault localization. Different measures perform better on different types of applications.

III. What is an explanation?

An adequate explanation to an output of an automated procedure is essential in many areas, including verification, planning, diagnosis, etc. It is clear that an explanation is essential in order to increase a user's confidence in the result or to determine whether there is a fault in the automated procedure (if the explanation does not make sense). It is less clear how to define what a "useful" explanation is. There have been a number of definitions of explanations over the years in various domains of computer science [28]–[30], philosophy [31] and statistics [32]. 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 [33] 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 going to loosely adopt the definition of explanations by Halpern and Pearl [34], which is based on their definition of actual causality [35]. Roughly speaking, Halpern and Pearl state that a good explanation gives an answer to the question *"why did this outcome occur"*, and is similar in spirit to DARPA's informal description. As we are not defining our setting in terms of actual causality, we are omitting 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.

A straightforward approach to computing an explanation consistent with Definition 1 would be to check all subsets of pixels of a given image for minimality and sufficiency for the DNN to classify the image. The run-time complexity of this approach, however, is exponential in the size of the image, and is hence infeasible for all but very tiny images. In Section IV we describe a different approach to computing an explanation and argue that it produces a good approximation for a precise explanation.

There exist other definitions of explanations for decisions of DNNs in the literature [8]. They are not used for presenting an explanation to the user, but rather as a theoretical model for ranking the pixels. We argue that our definition suits its purpose of explaining the DNN's decisions to the user, as it matches the intuition of what would constitute a good explanation and is consistent with the body of work on explanations in AI.

IV. Spectrum-Based Explanation (SBE) for DNNs

We propose a *lightweight black-box explanation technique* based on spectrum fault localization. In traditional software development, SBFL 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.

Constructing the test suite: SBFL 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 PROTOZOA algorithm.

Analyzing $T(x)$: 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 DNN's output. In the following, we will use SBFL measures to find a set of pixels that constitute an explanation of the DNN's output for x.

We use SBFL 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 $\langle a^i_{ep}, a^i_{ef}, a^i_{np}, a^i_{nf} \rangle$ as follows:

- a^i_{ep} stands for the number of mutants in $T(x)$ annotated with y in which p_i is not masked;
- a^i_{ef} stands for the number of mutants in $T(x)$ annotated with $\neg y$ in which p_i is not masked;
- a^i_{np} stands for the number of mutants in $T(x)$ annotated with y in which p_i is masked;
- a^i_{nf} stands for the number of mutants in $T(x)$ annotated with $\neg y$ in which p_i is masked.

Once we construct the vector $\langle a^i_{ep}, a^i_{ef}, a^i_{np}, a^i_{nf} \rangle$ for every pixel, we can apply SBFL measures discussed in Section II-B to rank the pixels in x for their importance regarding the DNN's output (the importance corresponds to the suspiciousness score computed by SBFL measures). A set of top-ranked pixels (10% of the pixels for most images) is provided to the user as a heuristic explanation of the decision of the DNN. This set is chosen 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.

A. Spectrum-based explanation (SBE) algorithm

We now present our algorithms for generating test suites and computing explanations. The computation of an SBE for a given DNN is described in detail in Algorithm 1. Given the DNN $\mathcal{N}$, a particular input x and a particular fault localization measure M, it synthesizes the subset of pixels $\mathcal{P}^{exp}$ that present an approximation of an explanation according to Definition 1.

Algorithm 1 starts by calling the $test\_inputs\_gen$ procedure 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 the set $T(x)$. Then, the algorithm computes the ranking of each pixel according to the specified measure M (Lines 3–5). Formulas for measures are as in Equation (2a)–(2d). The pixels are listed in the descending order of ranking (from high $value$ to low $value$) (Line 6).

Starting from Line 7 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 $y = \mathcal{N}[x]$. 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 SBFL 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

Algorithm 1 Spectrum Based Explanation for DNNs

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

```
1: $T(x) \leftarrow test\_inputs\_gen(\mathcal{N}, x)$
2: for each pixel $p_i \in \mathcal{P}$ do
3:     calculate $a^i_{ep}, a^i_{ef}, a^i_{np}, a^i_{nf}$ from $T(x)$
4:     $value_i \leftarrow M(a^i_{ep}, a^i_{ef}, a^i_{np}, a^i_{nf})$
5: end for
6: $pixel\_ranking \leftarrow$ pixels in $\mathcal{P}$ from high $value$ to low
7: $\mathcal{P}^{exp} \leftarrow \emptyset$
8: for each pixel $p_i \in pixel\_ranking$ do
9:     $\mathcal{P}^{exp} \leftarrow \mathcal{P}^{exp} \cup \{p_i\}$
10:    $x^{exp} \leftarrow$ mask pixels of $x$ that are not in $\mathcal{P}^{exp}$
11:    if $\mathcal{N}[x^{exp}] = \mathcal{N}[x]$ then
12:        return $\mathcal{P}^{exp}$
13:    end if
14: end for
```

our spectrum based 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).

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) 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 $(\sigma \cdot n)$ of pixels in x is masked and the resulting new input x' is added to $T(x)$ (Lines 4–5). Roughly speaking, if a current mutant is not classified as being the same as x, we decrease the fraction of masked pixels (by a pre-defined small number ϵ); if a current mutant is classified as x, we increase the fraction of masked pixels (by the same ϵ).

B. Relationship between SBE $\mathcal{P}^{exp}$ and Definition 1

Ranking the pixels using SBFL measures and then selecting the top-ranked pixels benefits from running time complexity that is linear in the size of the set $T(x)$ and the size of the image, and is hence much more efficient than the straightforward computation of all possible explanations as described in Section III. One of the reasons for this low complexity is that there can be many possible explanations (exponentially many, as any subset of pixels of the image can be an explanation), whereas we only need to provide one explanation to the user. It is also easy to see that the heuristic explanation that we provide is a sufficient set of pixels, since this is a stopping condition for adding pixels back to the image.

However, the set $\mathcal{P}^{exp}$ might not be minimal, and thus does not, strictly speaking, satisfy all conditions of Definition 1. The reason for possible non-minimality is that the pixels of x are

Algorithm 2 $test\_inputs\_gen(\mathcal{N}, x)$

INPUT: DNN $\mathcal{N}$, input x
OUTPUT: $T(x)$

```
1: $T(x) \leftarrow \emptyset$
2: $\sigma \leftarrow$ sample in the range $(0, 1)$
3: while $|T(x)| <$ some specified size $m$ do
4:     $x' \leftarrow$ randomly select and mask $\sigma \cdot n$ pixels in $x$
5:     $T(x) \leftarrow T(x) \cup \{x'\}$
6:     if $\mathcal{N}[x'] \neq \mathcal{N}[x]$ then
7:         $\sigma \leftarrow \max\{\sigma - \epsilon, 0\}$
8:     else
9:         $\sigma \leftarrow \min\{\sigma + \epsilon, 1\}$
10:    end if
11: end while
12: return $T(x)$
```

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 this is unlikely, as higher-ranked pixels tend to be more important to the correct classification than lower-ranked ones when using a good SBFL measure, based on emphirical evidence.

Finally, the SBEs we provide are clearly *not obvious* (defined in Section III), as the users do not know them in advance, thus fulfilling the condition of enriching the user's knowledge of the DNN's decisions.

V. Experimental Evaluation

In this section we describe the experimental evaluation of Protozoa. We start with showing that explanations generated by Protozoa match the human intuition about an explanation for a given classification of an image (Section V-B). We continue the evaluation by comparing Protozoa to SHAP (Section V-C). Since the degree of the alignment of our explanations with human intuition is expensive to quantify, we introduce two proxies for comparison and demonstrate experimental results on large sets of images. As SBFL measures inherently depend on the quality of the test suite, in Section V-D we present an evaluation demonstrating an impact of the size of the set of mutants $T(x)$ and the balance between passing and failing mutants in this set on the quality of explanations generated by Protozoa. Finally, in Section V-E we show that SBE can also be used to assess the quality of training of a given DNN: a non-intuitive explanation indicates that the training was not sufficient.

A. Setup

We implement the spectrum-based explanation (SBE) algorithm for DNNs presented in Section IV in the tool **Protozoa**.

The tool supports four fault localization measures, which are Tarantula, Zoltar, Ochiai and Wong-II. We evaluate the quality of explanations provided by PROTOZOA on image inputs from the ImageNet Large Scale Visual Recognition Challenge [36], which is the most sophisticated and comprehensive benchmark for the DNN recognition problem. Popular neural networks for ImageNet such as Xception [37], MobileNet [38], VGG16 [39] and InceptionV3 [40] have been integrated into the tool.

As SBE is very lightweight, we were able to conduct all experiments on a laptop with an Intel i7-7820HQ (8) running at 3.9 GHz and with 16 GB of memory. None of the experiments require a stronger machine.

We configure the heuristic tests generation in Algorithm 2 with $\sigma = \frac{1}{5}$ and $\epsilon = \frac{1}{6}$, and the size m of the test set $T(x)$ equal to $2{,}000$. These values have been chosen empirically and remain the same through all experiments. It is quite possible that they are not fine tuned to all image inputs, and that for some inputs increasing m or tuning σ and ϵ would produce a better explanation.

Tarantula is used as the default measure in PROTOZOA. Again, this was chosen empirically, as it seems that the explanations provided by Tarantula are the most intuitive for the majority of the images.

B. What does a well-trained DNN see?

We apply the SBE approach to Google's Xception model to illustrate how a state-of-the-art DNN makes decisions. A recent comparison of DNN models for ImageNet [41] suggests that Xception is one of the best models for this data set.

For each input image, the Xception DNN outputs a classification of the image out of $17{,}000$ classes, and we apply PROTOZOA to compute an explanation of this classification, that is, a subset of top-ranked pixels of the original image that explain why Xception classifies the image in the way it does.

Figure 1 exhibits a set of images and their corresponding explanations found by PROTOZOA. More results can be found at anonymized url https://github.com/theyoucheng/protozoa, and we encourage the reader to try PROTOZOA with different input images and neural network models.

Overall, the explanations computed by PROTOZOA match human intuition. It is straightforward for a human to identify which part of the image is supposed to trigger the DNN's decision. As we show later, the explanations can also be used to assess the quality of the training of the DNN. As Xception is a high-quality, well-trained DNN, the explanations for its decisions are highly consistent with our intuition, and, in particular, do not contain significant parts of the background, which should be irrelevant for the DNN's decision.

C. Quantitative evaluation

As this is the first paper that applies spectrum-based fault localization to explaining the outputs of deep neural networks, we focus on the feasibility and usefulness of the explanations, and less on possible performance optimizations.

We compare PROTOZOA with SHAP[2], the state-of-the-art machine learning tool to explain DNN outputs. Given a particular input image, SHAP assigns each of its pixels an *importance value*; higher values correspond to pixels that are more important for the DNN's output. The explanation then can be constructed by identifying the pixels that are top ranked. For the comparison between the tools, we replace the $pixel\_ranking$ in Algorithm 1 with the importance ranking computed by SHAP.

We use MobileNet as the DNN model that is to be explained: it is is nearly as accurate as VGG16, while being 27 times faster [38]. Moreover, we have observed that the explanations generated for several mainstream ImageNet models, including Xception, MobileNet and VGG16, are largely consistent.

It is challenging to evaluate the quality of DNN explanations, owing to the lack of an objective measure. As we saw in Section V-B, the quality of explanations is a matter of perception by humans. To compare several explanations for DNN outputs automatically at a large scale, we need computable metrics. We design two proxies for this purpose: (1) the size of generated explanations, and (2) the generation of adversarial examples.

Size of explanations: 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. When comparing explanations, the ranking for PROTOZOA is computed as described in the algorithm; for SHAP, we use the importance values of the pixels. We define the size of the explanation as $\frac{|\mathcal{P}^{exp}|}{|\mathcal{P}|}$. Intuitively, the smaller this size is, the more accurately we captured the decision process of the DNN, hence smaller explanations are considered better.

Figure 2 gives the comparison with respect to the size of generated explanations between our SBE approach and SHAP. For each point in Figure 2, the position on the x-axis indicates 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 sizes. Figure 2 contains the SBE results for four SBE measures (Ochiai, Zoltar, Tarantula and Wong-II) that are used for ranking; the blue line for PROTOZOA represents the explanation with smallest size among the four measures.

The data in Figure 2 allows us to make the following observations.

- Using spectrum-based ranking for explanations is significantly better in terms of the size of the explanation compared to SHAP on the images in ImageNet.
- Except for Wong-II, the results produced by spectrum-based measures are very close to each other; on the other hand, no single measure consistently outperforms the others on all input images; hence PROTOZOA, which chooses the smallest explanation for each image, outperforms all individual measures.

Figure 3 gives an example of an input image ("original image", depicting a raccoon) and the explanations produced when using four SBFL measures and when using SHAP. We can

[2]https://github.com/slundberg/shap

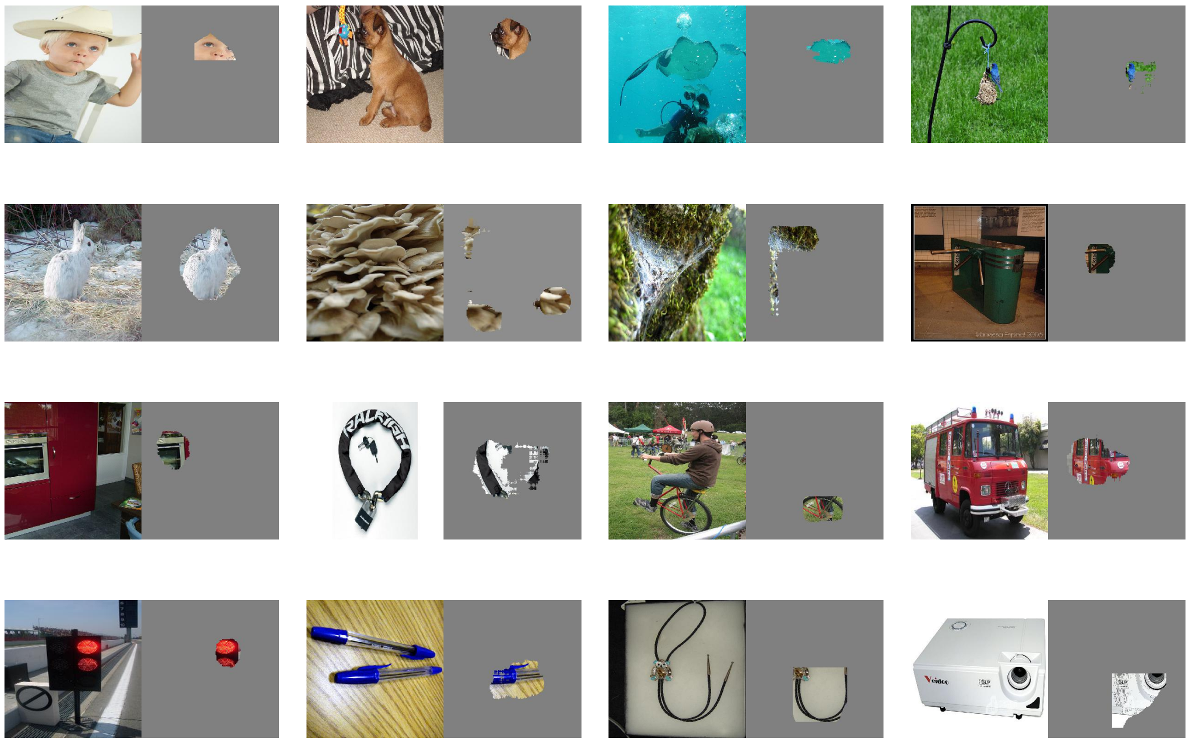

Fig. 1: Input images and explanations for Xception model (ImageNet validation data set)

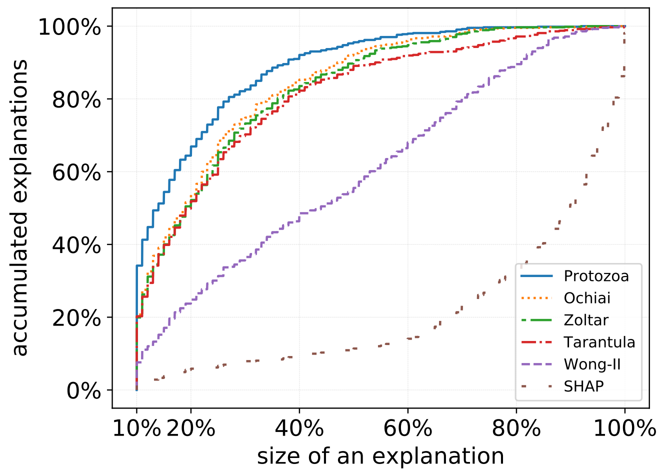

Fig. 2: Comparison between our SBE approach and SHAP in the size of generated explanations (MobileNet, ImageNet validation data set)

see that the explanation based on SHAP's importance values classifies many background pixels as important, hence resulting in a large explanation. By contrast, Tarantula top-ranks the pixels that belong to the raccoon's head (and are presumably the most important for correct classification), resulting in a much smaller explanation. On this image, Ochiai and Zoltar produce similar explanations (better than SHAP, but worse than Tarantula), and Wong-II, while localizing a part of the raccoon's image, gives a high ranking to more background pixels than any of the other SBFL measures.

Another observation that is illustrated well by Figure 3 and that holds for almost all images in our evaluation, is that explanations based on SHAP's importance values tend to resemble low-resolution variants of the original images. They consist of sets of pixels spread across the entire image, and include a lot of background. By contrast, our explanations focus on one area that is crucial for classifying the image.

Generation of adversarial examples: Adversarial examples [42] are a major safety concern for DNNs. An adversarial example is defined to be a perturbed input image that is a) very close to an image in the data set and that is b) classified with a different label by the DNN. In this section, we use adversarial examples as a proxy to compare the effectiveness of spectrum-based explanations and SHAP. In particular, following the ordering of pixels according to their ranking (from the SBE approach or SHAP), we change the original image pixel by pixel starting from the top-ranked ones until the DNN changes its classification, i.e., an adversarial example is found. We limit the number of changed pixels to 10%. We then record the number of pixels changed (normalized over the total number of pixels) in the adversarial example. In our setup, changing a pixel means assigning it black color.

There is a significant body of research dedicated to the efficient generation of adversarial examples [43]–[45], and

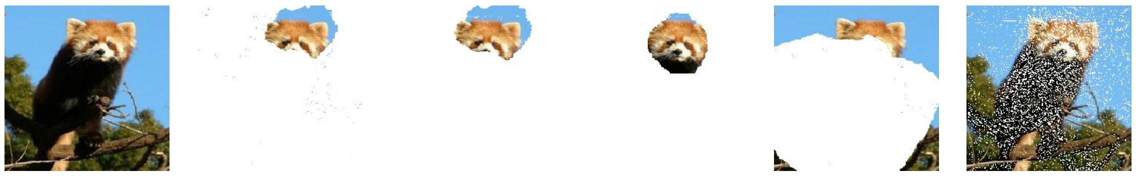

Fig. 3: Explanations obtained using four SBE measures and SHAP

we do not attempt to compete with the existing specialised methods. Notably, adversarial examples can be generated by changing a single pixel only [46]. In our setup, the changes to pixels are inherently pessimistic (in other words, there might be another color that leads to more a efficient generation of adversarial examples). We remind the reader that our framework for generating adversarial examples is solely used as a proxy to assess the quality of explanations of PROTOZOA and SHAP.

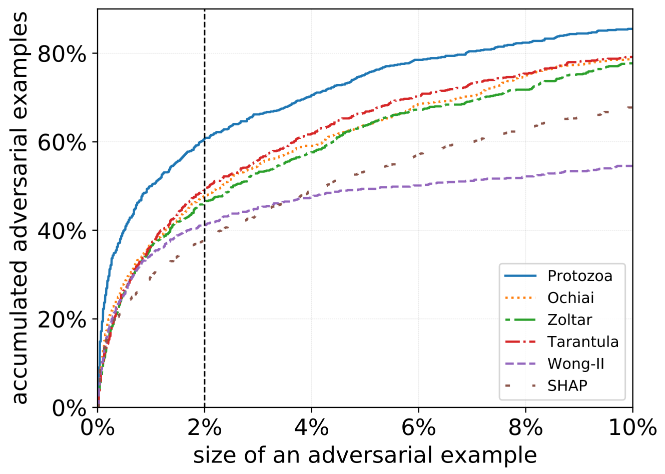

Fig. 4: Comparison between our SBE approach and SHAP in adversarial quality (MobileNet, ImageNet validation data set)

Figure 4 provides the comparison between the SBE measures and SHAP for guiding the generation of adversarial examples. Figure 5 gives an example of this method: for the input image of a salt shaker, we provide the adversarial examples generated by the SBFL measures and by SHAP. It is easy to see that, while all generated images still look like a salt shaker, the one generated using the Tarantula ranking produces an adversarial example that is closest to the original image.

On average, a DNN's output can be changed by modifying a very small number of pixels in the image, much smaller than the number of pixels needed for a correct classification of the image. This explains why the gap between different approaches in Figure 4 is significantly smaller than in Figure 2. Yet, it is still clear that SHAP's ranking yields a much lower number of adversarial examples than almost all SBFL measures, and hence also PROTOZOA. Moreover, no single measure consistently outperforms the others on all input images (note that the performance of Wong-II significantly degrades after changing more than 2% of pixels); hence PROTOZOA, which chooses the smallest modification for each image, outperforms all individual measures significantly.

The experiment shows that the ranking computed by the SBFL measures is more efficient than the one computed by SHAP for guiding the generation of adversarial examples. This result is consistent with the results in Figure 2.

D. Tuning the parameters in Algorithm 2

In this section we study the effect of changing the parameters in Algorithm 2, and, specifically, the size of the set of mutant images $T(x)$ and the parameters σ and ϵ that are used for generating passing and failing mutants. We show that, as expected, the quality of explanations improves with a bigger set of tests $T(x)$; however, changing the balance between the passing and the failing mutants in $T(x)$ does not seem to have a significant effect on the results.

We conduct two experiments. In the first experiment, we study the effect of changing the size of $T(x)$ by computing the ranking using the different mutant sets. In the original setup, $|T(x)| = 2{,}000$. We generate a smaller set $T'(x)$ of size 200, and we compare the explanations obtained when using $T(x)$ to the ones obtained when using $T'(x)$. In Figure 6, we show the average size of the explanations for different SBFL measures and sets of mutant images of size 200 and 2,000.

As expected, the quality of SBEs improves, meaning they have fewer pixels, when more test inputs are used as spectra in Algorithm 1. This suggests that the effort of using a large set of test inputs $T(x)$ is rewarded with a high quality of the generated explanations for the decisions of the DNN. We remark that this observation is hardly surprising, and is consistent with prior experience applying spectrum-based fault localization measures to traditional software.

In Figure 7 we record the running time of PROTOZOA for different $|T(x)|$ and compare it to the running time of SHAP. The running time of PROTOZOA is separated into two parts: the time taken for the execution of the test set $T(x)$ (Algorithm 2) and the time taken for the subsequent computation of the ranked list and extracting an explanation (Algorithm 1). It is easy to see that almost the whole execution time of PROTOZOA is dedicated to the execution of $T(x)$. When comparing the explanation extraction only, PROTOZOA is more efficient than SHAP. Hence, if the set $T(x)$ is computed in advance or is given to PROTOZOA as an input, the computation of SBE is very lightweight. Another alternative for improving the running time is to first execute PROTOZOA with a small set $T'(x)$ (of

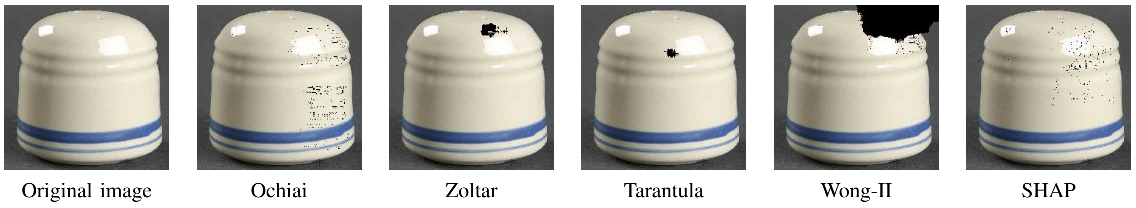

Fig. 5: Adversarial examples (**"saltshake"** $\rightarrow$ **"thimble"**) generated using four SBE measures and using SHAP

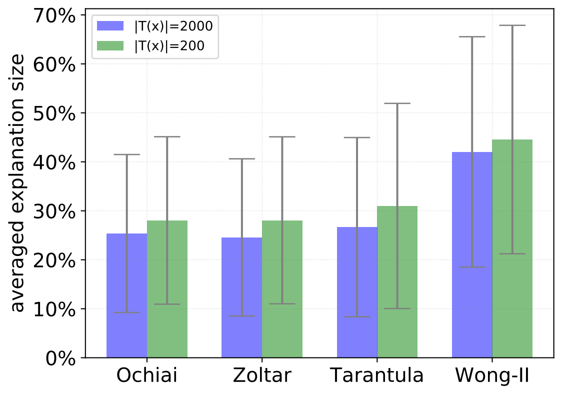

Fig. 6: The size of the explanation for four measures when $|T(x)|$ varies (MobileNet, ImageNet validation data set)

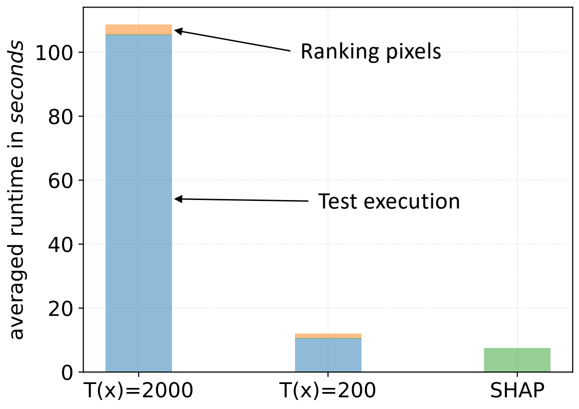

Fig. 7: Runtime comparison between different configurations of PROTOZOA and SHAP (MobileNet, ImageNet validation dataset)

200 tests), and to generate a large $T(x)$ only if the explanation is low quality.

When SBFL measures are applied to software, the quality of the ranking is known to depend on the balance between passing and failing traces in the test suite. In our setting, this is the balance is between the tests labeled with "y" and with "$\neg y$" in $T(x)$. That balance is controlled by the parameters σ and ϵ. We test the dependence of the quality of SBEs on this balance between the tests directly by designing the following two types of test suites (both with 2,000 tests):

- the "Type-y" kind of $T(x)$ is generated by adding an additional set of tests annotated with "y"; and
- the "Type-not-y" kind of $T(x)$ is generated by adding an additional set of tests annotated with "$\neg y$".

Thus, instead of relying on σ and ϵ to provide a balanced set of tests, we tip the balance off intentionally. We then run PROTOZOA with these two types of biased sets of tests.

Figure 8 gives the sizes of explanations for the two types of sets of tests. It is easy to see that the PROTOZOA algorithm is remarkably robust with respect to the balance between the different types of tests in $T(x)$ (as the columns are of roughly equal height). Again, Wong-II stands out and appears to be more sensitive to the ratio of failing/passing tests in $T(x)$.

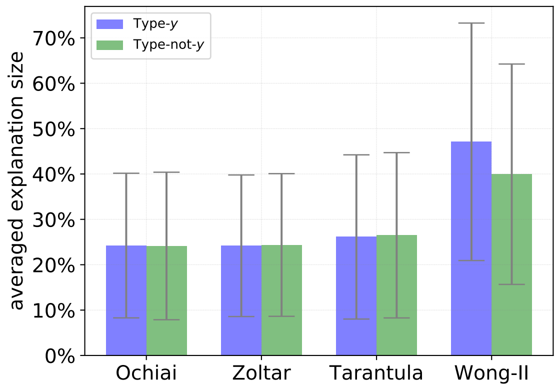

Fig. 8: The explanation size of SBEs with different types of $T(x)$ (MobileNet, ImageNet validation data set)

E. Using explanations to assess the progress of training of DNNs

An important use-case of explanations of DNN outputs is assessing the adequacy of training of the DNN. To demonstrate this, we have trained a DNN on the CIFAR-10 data set [47]. We apply PROTOZOA after each iteration of the training process to the intermediate DNN model. In Figure 9 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 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

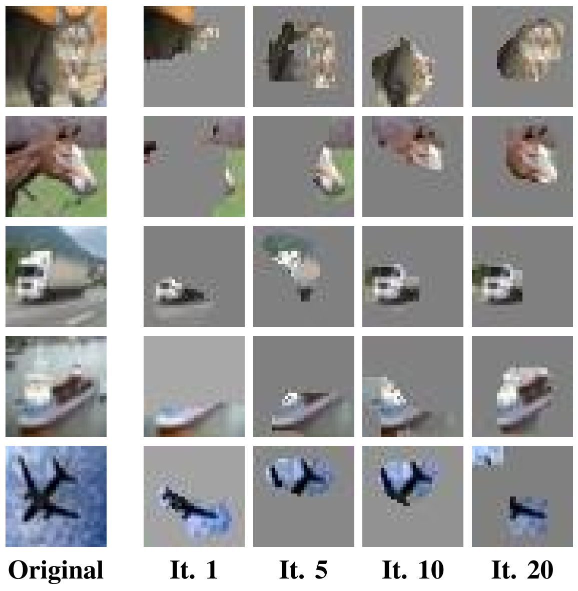

Fig. 9: Explanations of the DNN at different training stages: the 1st column are the original images and each later column represents explanations from an iteration in the training (CIFAR-10 validation data set)

also observed that the DNN's feature learning is not always monotonic, as demonstrated in the bottom row of Figure 9: after the 10th iteration, explanations for the DNN's classification of an input image as an "airplane" drift from 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 SBE algorithm).

The explanations generated by PROTOZOA may thus be useful for assessing the adequacy of the DNN training; they may enable checks whether the DNN is aligned with the developer's intent when training the neural network. The explanations can be used as a stopping condition for the training process: training is finished when the explanations align with our intuition.

VI. Threats to Validity

Lack of ground truth: When evaluating the generated explanations, there is no ground truth to compare. Ultimately, we use two proxies, the size of the explanation and the effort required for generating adversarial examples.

Selection of the dataset: In this paper, we focus on the image recognition problem for high-resolution color images and collect most of the experimental results using the ImageNet data set. Small benchmarks and problems may have their own features that differ from what we report in this paper. It is known that, in traditional software, the performance of different spectrum-based measures can vary dramatically given the benchmark used. SHAP has been applied to DNNs with non-image input.

Selection of SBFL measures: We have only evaluated four spectrum-based 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.

Adversarial example generation algorithm: There is a variety of methods to generate adversarial examples, including sophisticated optimization algorithms. Instead, as a proxy to evaluate the effectiveness of explanations from PROTOZOA and SHAP, we adopt a simple method that blacks out selected pixels of the original image. A more sophisticated algorithm might yield different results, and might favor the explanations generated by SHAP.

VII. Related Work

This work connects two seemingly distinct research topics: spectrum-based fault localization and explainable AI. We briefly summarize related research from software engineering and machine learning.

Machine learning: Explanation or interpretation of deep learning is a very active area in machine learning. Explanations of trained models is done by visualising hidden neurons, ranking input dimensions and other methods. LIME [9] interprets model predictions by locally approximating the model around a given prediction. Based on LIME and a few other methods [48]–[51], SHAP [8] suggests a general additive model and defines the importance value of each pixel as its Shapley value in a multi-player cooperative game, where each player represents a pixel. SHAP is the most up-to-date ranking method, and is compared with our method in the paper. The core of our proposed SBE method is also pixel ranking, however, our ranking uses the suspiciousness score computed by SBFL measures.

Fault localization: Our SBE approach is closely related to fault localization. Besides the spectrum-based fault localization [12]–[26] discussed earlier in this paper, there are a large number of further fault localization methods, e.g., model based [52], [53], slice based [54], interaction driven [55], [56], similarity aware [57], semantic fault localization [58] and a number of others [59]–[62]. As indicated in work like [27], [63]–[65], there may be merit in combining different fault localization methods. Meanwhile, work like [66]–[69] focus on better constructing or optimizing the test suite for fault localization. Fault localization is related to automating fixes for programs [70], [71]. Genetic programming can be used to improve SBFL [72], [73].

Software engineering for AI: There is a broad body of work on applying software engineering research to deep learning, and this paper is aligned with this idea. Among them, DeepFault [74] is closest to ours. It applies spectrum-based fault localization to identify suspicious neurons that are responsible for inadequate DNN results; it is a white-box method and it was only tested on small benchmarks such as MNIST [75] and

CIFAR-10 [47]. In [76], symbolic execution is used to find important pixels for the MNIST data set.

In [77]–[79], multiple structural test coverage criteria have been proposed for DNNs with the goal to support testing. The growing safety concern in deep learning based autonomous systems drives the work on testing methods for the DNN component [80]–[82]. A number of testing approaches have been adopted for testing DNN models [83]–[89]. The most popular library to implement DNNs is TensorFlow [90] and an empirical study of TensorFlow program bugs can be found in [91].

VIII. Conclusions

This paper advocates the application of spectrum-based fault localization for the generation of explanations of the output of neural networks. We have implemented this idea in the tool PROTOZOA. Experimental results using advanced deep learning models and comparing our tool with SHAP confirm that our spectrum-based approach to explanations is able to deliver good explanations for DNN's outputs at low computational cost.

This work can be extended in numerous directions. We have demonstrated that applying well-known spectrum-based fault localization measures is useful for providing explanations of the output of a DNN. As these measures are based on statistical correlation, it may be the case that measures that work well for fault localization in general software are inadequate for DNNs; it may be worthwhile to investigate new measures that are specialized for DNNs. Our work uses random mutation to produce the set of test inputs. While this is an efficient approach to generate the test inputs, it may be possible to obtain better explanations by using a more sophisticated method for generating the test inputs; an option are white-box fuzzers such as AFL.

Another direction is to extend the explanation-based approach to other areas where DNNs are used. While the type of explanations we construct in this paper works well for images, it might be that for other input types other techniques will be more suitable. Finally, as our algorithm is agnostic to the structure of the DNN, it applies immediately to DNNs with state, say recurrent neural networks for video processing. Future work could benchmark the quality of explanations generated for this use case.

References

[1] I. Rahwan, M. Cebrian, N. Obradovich, J. Bongard, J.-F. Bonnefon, C. Breazeal, J. W. Crandall, N. A. Christakis, I. D. Couzin, M. O. Jackson *et al.*, "Machine behaviour," *Nature*, vol. 568, no. 7753, p. 477, 2019.

[2] AI HLEG, "Ethics guidelines for trustworthy AI." [Online]. Available: https://ec.europa.eu/futurium/en/ai-alliance-consultation

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[4] C. Ziegler, "A Google self-driving car caused a crash for the first time," *The Verge*, 2016. [Online]. Available: https://www.theverge.com/2016/2/29/11134344/google-self-driving-car-crash-report

[5] F. Lambert, "Understanding the fatal Tesla accident on autopilot and the NHTSA probe," *Electrek, July*, vol. 1, 2016. [Online]. Available: https://electrek.co/2016/07/01/understanding-fatal-tesla-accident-autopilot-nhtsa-probe/

[6] D. Gunning, "Explainable artificial intelligence (XAI)," *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2017.

[7] C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev, "The building blocks of interpretability," *Distill*, 2018, https://distill.pub/2018/building-blocks.

[8] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4765–4774. [Online]. Available: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

[9] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you? Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1135–1144.

[10] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *Proceedings of the 34th International Conference on Machine Learning*, vol. 70. JMLR.org, 2017, pp. 3145–3153.

[11] A. Datta, S. Sen, and Y. Zick, "Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems," in *2016 IEEE symposium on security and privacy (SP)*. IEEE, 2016, pp. 598–617.

[12] L. Naish, H. J. Lee, and K. Ramamohanarao, "A model for spectra-based software diagnosis," *ACM Transactions on software engineering and methodology (TOSEM)*, vol. 20, no. 3, p. 11, 2011.

[13] D. Lo, L. Jiang, A. Budi *et al.*, "Comprehensive evaluation of association measures for fault localization," in *2010 IEEE International Conference on Software Maintenance*. IEEE, 2010, pp. 1–10.

[14] L. Lucia, D. Lo, L. Jiang, F. Thung, and A. Budi, "Extended comprehensive study of association measures for fault localization," *Journal of software: Evolution and Process*, vol. 26, no. 2, pp. 172–219, 2014.

[15] D. Landsberg, H. Chockler, D. Kroening, and M. Lewis, "Evaluation of measures for statistical fault localisation and an optimising scheme," in *International Conference on Fundamental Approaches to Software Engineering*. Springer, 2015, pp. 115–129.

[16] W. E. Wong, V. Debroy, and B. Choi, "A family of code coverage-based heuristics for effective fault localization," *Journal of Systems and Software*, vol. 83, no. 2, pp. 188–208, 2010.

[17] L. Zhang, L. Yan, Z. Zhang, J. Zhang, W. Chan, and Z. Zheng, "A theoretical analysis on cloning the failed test cases to improve spectrum-based fault localization," *Journal of Systems and Software*, vol. 129, pp. 35–57, 2017.

[18] A. Perez, R. Abreu, and A. van Deursen, "A test-suite diagnosability metric for spectrum-based fault localization approaches," in *Proceedings of the 39th International Conference on Software Engineering*. IEEE Press, 2017, pp. 654–664.

[19] B. Jiang, W. Chan, and T. Tse, "On practical adequate test suites for integrated test case prioritization and fault localization," in *11th International Conference on Quality Software*. IEEE, 2011, pp. 21–30.

[20] R. Santelices, J. A. Jones, Y. Yu, and M. J. Harrold, "Lightweight fault-localization using multiple coverage types," in *Proceedings of the 31st International Conference on Software Engineering*. IEEE Computer Society, 2009, pp. 56–66.

[21] R. Feldt, S. Poulding, D. Clark, and S. Yoo, "Test set diameter: Quantifying the diversity of sets of test cases," in *International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2016, pp. 223–233.

[22] W. E. Wong, R. Gao, Y. Li, R. Abreu, and F. Wotawa, "A survey on software fault localization," *IEEE Transactions on Software Engineering*, vol. 42, no. 8, pp. 707–740, 2016.

[23] A. Gonzalez-Sanchez, "Automatic error detection techniques based on dynamic invariants," *M.S. Thesis, Delft University of Technology, The Netherlands*, 2007.

[24] A. Ochiai, "Zoogeographic studies on the soleoid fishes found in Japan and its neighbouring regions," *Bulletin of Japanese Society of Scientific Fisheries*, vol. 22, pp. 526–530, 1957.

[25] J. A. Jones and M. J. Harrold, "Empirical evaluation of the Tarantula automatic fault-localization technique," in *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*. ACM, 2005, pp. 273–282.

[26] W. E. Wong, Y. Qi, L. Zhao, and K.-Y. Cai, "Effective fault localization using code coverage," in *31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*, vol. 1. IEEE, 2007, pp. 449–456.

[27] D. Zou, J. Liang, Y. Xiong, M. D. Ernst, and L. Zhang, "An empirical study of fault localization families and their combinations," *IEEE Transactions on Software Engineering*, 2019.

[28] U. Chajewska and J. Y. Halpern, "Defining explanation in probabilistic systems," in *UAI '97: Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 1997, pp. 62–71.

[29] P. Gärdenfors, *Knowledge in Flux*. MIT Press, 1988.

[30] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[31] C. G. Hempel, *Aspects of Scientific Explanation*. Free Press, 1965.

[32] W. C. Salmon, *Four Decades of Scientific Explanation*. University of Minnesota Press, 1989.

[33] D. Gunning, "Explainable artificial intelligence (XAI) – program information," https://www.darpa.mil/program/explainable-artificial-intelligence, 2017, Defense Advanced Research Projects Agency.

[34] J. Y. Halpern and J. Pearl, "Causes and explanations: a structural-model approach. Part II: Explanations," vol. 56, no. 4, 2005.

[35] ——, "Causes and explanations: a structural-model approach. Part I: Causes," vol. 56, no. 4, 2005.

[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[37] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[38] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[39] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[40] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[41] [Online]. Available: https://keras.io/applications/

[42] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *In ICLR*. Citeseer, 2014.

[43] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[44] I. Goodfellow, N. Papernot, S. Huang, Y. Duan, P. Abbeel, and J. Clark, "Attacking machine learning with adversarial examples," Mar 2017. [Online]. Available: https://blog.openai.com/adversarial-example-research/

[45] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Security and Privacy (SP), IEEE Symposium on*, 2017, pp. 39–57.

[46] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, 2019.

[47] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[48] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *International Conference on Machine Learning*, 2017.

[49] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLoS ONE*, vol. 10, no. 7, 2015.

[50] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *Proceedings of Machine Learning Research 70:3145-3153*, 2017.

[51] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra, "Grad-CAM: Why did you say that? Visual explanations from deep networks via gradient-based localization," in *NIPS 2016 Workshop on Interpretable Machine Learning in Complex Systems*, 2016.

[52] G. Birch, B. Fischer, and M. Poppleton, "Fast test suite-driven model-based fault localisation with application to pinpointing defects in student programs," *Software & Systems Modeling*, pp. 1–27, 2017.

[53] E. H. da S. Alves, L. C. Cordeiro, and E. B. de L. Filho, "A method to localize faults in concurrent C programs," *Journal of Systems and Software*, vol. 132, pp. 336–352, 2017.

[54] E. Alves, M. Gligoric, V. Jagannath, and M. d'Amorim, "Fault-localization using dynamic slicing and change impact analysis," in *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE Computer Society, 2011, pp. 520–523. [Online]. Available: http://dx.doi.org/10.1109/ASE.2011.6100114

[55] X. Li, M. dAmorim, and A. Orso, "Iterative user-driven fault localization," in *Haifa Verification Conference*. Springer, 2016, pp. 82–98.

[56] D. Hao, L. Zhang, T. Xie, H. Mei, and J.-S. Sun, "Interactive fault localization using test information," *Journal of Computer Science and Technology*, vol. 24, no. 5, pp. 962–974, 2009.

[57] D. Hao, L. Zhang, Y. Pan, H. Mei, and J. Sun, "On similarity-awareness in testing-based fault localization," *Automated Software Engineering*, vol. 15, no. 2, pp. 207–249, 2008.

[58] M. Christakis, M. Heizmann, M. N. Mansur, C. Schilling, and V. Wüstholz, "Semantic fault localization and suspiciousness ranking," in *25th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, vol. 25. Springer, 2019.

[59] P. S. Kochhar, X. Xia, D. Lo, and S. Li, "Practitioners' expectations on automated fault localization," in *Proceedings of the 25th International Symposium on Software Testing and Analysis*. ACM, 2016, pp. 165–176.

[60] X. Xia, L. Bao, D. Lo, and S. Li, "Automated debugging considered harmful: A user study revisiting the usefulness of spectra-based fault localization techniques with professionals using real bugs from large systems," in *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2016, pp. 267–278.

[61] J. Zhou, H. Zhang, and D. Lo, "Where should the bugs be fixed? more accurate information retrieval-based bug localization based on bug reports," in *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, 2012, pp. 14–24.

[62] C.-P. Wong, Y. Xiong, H. Zhang, D. Hao, L. Zhang, and H. Mei, "Boosting bug-report-oriented fault localization with segmentation and stack-trace analysis," in *2014 IEEE International Conference on Software Maintenance and Evolution*. IEEE, 2014, pp. 181–190.

[63] D. L. Lucia and X. Xia, "Fusion fault localizers," in *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, 2014, pp. 127–138.

[64] S. Wang and D. Lo, "Version history, similar report, and structure: Putting them together for improved bug localization," in *Proceedings of the 22nd International Conference on Program Comprehension*. ACM, 2014, pp. 53–63.

[65] T.-D. B. Le, R. J. Oentaryo, and D. Lo, "Information retrieval and spectrum based bug localization: better together," in *Foundations of Software Engineering*. ACM, 2015, pp. 579–590.

[66] D. Hao, T. Xie, L. Zhang, X. Wang, J. Sun, and H. Mei, "Test input reduction for result inspection to facilitate fault localization," *Automated software engineering*, vol. 17, no. 1, p. 5, 2010.

[67] D. Landsberg, Y. Sun, and D. Kroening, "Optimising spectrum based fault localisation for single fault programs using specifications." in *FASE*, 2018, pp. 246–263.

[68] J. Campos, R. Abreu, G. Fraser, and M. d'Amorim, "Entropy-based test generation for improved fault localization," in *Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering*. IEEE Press, 2013, pp. 257–267.

[69] A. Christi, M. L. Olson, M. A. Alipour, and A. Groce, "Reduce before you localize: Delta-debugging and spectrum-based fault localization," in *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 2018, pp. 184–191.

[70] M. Böhme, E. O. Soremekun, S. Chattopadhyay, E. Ugherughe, and A. Zeller, "Where is the bug and how is it fixed? An experiment with practitioners," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ACM, 2017, pp. 117–128.

[71] K. Liu, A. Koyuncu, T. F. Bissyandé, D. Kim, J. Klein, and Y. Le Traon, "You cannot fix what you cannot find! An investigation of fault localization bias in benchmarking automated program repair systems," in *Proceedings of the 12th IEEE International Conference on Software Testing, Verification and Validation*. IEEE, 2019.

[72] J. Sohn and S. Yoo, "FLUCCS: Using code and change metrics to improve fault localisation," in *Proceedings of International Symposium on Software Testing and Analysis*, ser. ISSTA 2017, Jul. 2017, pp. 273–283.

[73] K. Choi, J. Sohn, and S. Yoo, "Learning fault localisation for both humans and machines using Multi-Objective GP," in *Proceedings of the 10th International Symposium on Search Based Software Engineering*, ser. SSBSE 2018, 2018, pp. 349–355.

[74] S. Gerasimou, H. Eniser, and A. Sen, "DeepFault: Fault localization for deep neural networks," in *22nd International Conference on Fundamental Approaches to Software Engineering*. Springer, 2019.

[75] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[76] D. Gopinath, K. Wang, M. Zhang, C. S. Pasareanu, and S. Khurshid, "Symbolic execution for deep neural networks," *arXiv preprint arXiv:1807.10439*, 2018.

[77] K. Pei, Y. Cao, J. Yang, and S. Jana, "DeepXplore: Automated whitebox testing of deep learning systems," in *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017, pp. 1–18.

[78] L. Ma, F. Juefei-Xu, J. Sun, C. Chen, T. Su, F. Zhang, M. Xue, B. Li, L. Li, Y. Liu, J. Zhao, and Y. Wang, "DeepGauge: Comprehensive and multi-granularity testing criteria for gauging the robustness of deep learning systems," in *Automated Software Engineering (ASE)*. ACM, 2018, pp. 120–131.

[79] Y. Sun, X. Huang, D. Kroening, J. Sharp, M. Hill, and R. Ashmore, "Testing deep neural networks," *arXiv preprint arXiv:1803.04792*, 2018.

[80] Y. Tian, K. Pei, S. Jana, and B. Ray, "DeepTest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th International Conference on Software Engineering*. ACM, 2018, pp. 303–314.

[81] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "DeepRoad: GAN-based metamorphic autonomous driving system testing," in *Automated Software Engineering (ASE), 33rd IEEE/ACM International Conference on*, 2018.

[82] S. Lan, C. Huang, Z. Wang, H. Liang, W. Su, and Q. Zhu, "Design automation for intelligent automotive systems," in *2018 IEEE International Test Conference (ITC)*. IEEE, 2018, pp. 1–10.

[83] L. Ma, F. Zhang, M. Xue, B. Li, Y. Liu, J. Zhao, and Y. Wang, "Combinatorial testing for deep learning systems," *arXiv preprint arXiv:1806.07723*, 2018.

[84] L. Ma, F. Zhang, J. Sun, M. Xue, B. Li, F. Juefei-Xu, C. Xie, L. Li, Y. Liu, J. Zhao *et al.*, "DeepMutation: Mutation testing of deep learning systems," in *Software Reliability Engineering, IEEE 29th International Symposium on*, 2018.

[85] Y. Sun, M. Wu, W. Ruan, X. Huang, M. Kwiatkowska, and D. Kroening, "Concolic testing for deep neural networks," in *Automated Software Engineering (ASE), 33rd IEEE/ACM International Conference on*, 2018.

[86] S. Udeshi, P. Arora, and S. Chattopadhyay, "Automated directed fairness testing," in *Automated Software Engineering (ASE), 33rd IEEE/ACM International Conference on*, 2018.

[87] J. Wang, J. Sun, P. Zhang, and X. Wang, "Detecting adversarial samples for deep neural networks through mutation testing," *arXiv preprint arXiv:1805.05010*, 2018.

[88] J. Wang, G. Dong, J. Sun, X. Wang, and P. Zhang, "Adversarial sample detection for deep neural network through model mutation testing," in *Proceedings of the 41st International Conference on Software Engineering*. ACM, 2019.

[89] J. Kim, R. Feldt, and S. Yoo, "Guiding deep learning system testing using surprise adequacy," *arXiv preprint arXiv:1808.08444*, 2018.

[90] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "TensorFlow: a system for large-scale machine learning," in *OSDI*, vol. 16, 2016, pp. 265–283.

[91] Y. Zhang, Y. Chen, S.-C. Cheung, Y. Xiong, and L. Zhang, "An empirical study on TensorFlow program bugs," in *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2018.