Preprint: Chandrasekaran, J., Lei, Y., Kacker, R., & Kuhn, D. R. (2021, April). A Combinatorial Approach to Explaining Image Classifiers. *IEEE International Conference on Software Testing, Verification and Validation Workshops* (ICSTW) (pp. 35-43). IEEE.

A Combinatorial Approach to Explaining Image Classifiers

Jaganmohan Chandrasekaran
Department of Computer Science & Engineering
The University of Texas at Arlington
Arlington, USA
jaganmohan.chandrasekaran@mavs.uta.edu

Yu Lei
Department of Computer Science & Engineering
The University of Texas at Arlington
Arlington, USA
ylei@cse.uta.edu

Raghu Kacker, D. Richard Kuhn
Information Technology Lab
National Institute of Standards and Technology
Gaithersburg, USA
{raghu.kacker, d.kuhn}@nist.gov

***Abstract*—Machine Learning (ML) models, a core component to artificial intelligence systems, often come as a black box to the user, leading to the problem of interpretability. Explainable Artificial Intelligence (XAI) is key to providing confidence and trustworthiness for machine learning-based software systems. We observe a fundamental connection between XAI and software fault localization. In this paper, we present an approach that uses BEN, a combinatorial testing-based software fault localization approach, to produce explanations for decisions made by ML models.**

Keywords— explainability, deep learning, software testing, debugging DNN models, explainable AI, combinatorial testing, Image classifiers, model-agnostic, counterfactual explanation, instance-level explanations

I. Introduction

Artificial Intelligence (AI) based software systems are increasingly adopted in safety-critical domains, e.g. medical imaging and autonomous driving. At the core of AI-based software systems is a machine learning (ML) model that is used to perform tasks such as classification and prediction. The ML models used in such tasks are black box in nature, i.e., the reasoning behind their decision is typically not known to the user. Using a black box model in AI software systems could compromise trustworthiness and create problems such as racial and gender bias. There is an urgent need to provide explanations for the decisions made by AI-based software systems.

Explainable Artificial Intelligence (XAI) focuses on creating approaches and tools that can automatically provide explanations for the decisions made by ML models [21]. In particular, XAI tries to answer the following two questions: Why does the model make a particular decision? What are the major factors that contribute to the decision? XAI has attracted a lot of interest from both academia and industry in the past few years. Providing explanations allows a model to be interpreted, which is key to acceptance of AI technologies. Furthermore, information gathered from an interpretable model can help engineers determine the cause of incorrect decisions.

There are two types of explanations for AI decisions. *Local explanations* are created to explain a specific decision, whereas *global explanations* are created to explain an entire model. In this paper we present an approach that creates local explanations using the counterfactual approach, which human factor studies have shown to be highly effective for explanation [29]. A counterfactual approach tries to identify a minimum set of features that, if removed, would cause a different decision to be made [24].

The key insight is that from an abstract perspective, producing a counterfactual explanation for a local decision made by an ML model is similar to the fault localization problem [30][31]. In fault localization, given a failing scenario, a software developer identifies which part of the input that causes the failure. Similarly, in XAI, given a decision made by an ML model, we identify features that causes the decision, in the sense that if these features are removed, then the decision would be different.

Specifically, we explore the use of a combinatorial testing-based fault localization approach called BEN to produce counterfactual explanations for image classifiers. Given a t-way test set, BEN identifies a failure-inducing (or inducing) combination that causes every test (for a deterministic system) containing the combination to fail and that is as small as possible [8]. We apply BEN to quickly identify a minimal subset of features in an image that, if removed, would result in a different classification.

Assume that a model M produces a classification X for an input image I. To produce a counterfactual explanation for this classification result, we first perform segmentation on image I. In image segmentation, various algorithms are used to assign a class to each pixel of an image. For example, in a street scene, boundary detection and other algorithms may identify classes “sign”, “human”, “car”, etc., and each pixel of the image is associated with one of the classes. The segmentation process may be applied at a more granular level to identify parts of objects. Each segment is modeled as a Boolean parameter. We build a 2-way test set for these parameters. Each test can be used to derive a test image from the original image, i.e. image I. A segment is masked in the test image if the corresponding parameter is true in the test; otherwise, a segment is retained without modification.

The notion of test execution is mapped to image classification in the following sense. If a test image is classified by model M differently than the original image, the corresponding test execution is considered to be failing. Otherwise, the corresponding test execution is considered to be passing. The 2-way test set with execution statuses is then fed to BEN to identify inducing combinations. In the identification process, BEN could generate additional tests, which can be executed in the same manner. That is, for each additional test, a test image is first derived and then classified using model M to determine its execution status.

Finally, each inducing combination identified by BEN is used to derive an image that produces a different classification. This image serves as a counterfactual explanation for the original classification X.

We report an experimental evaluation of our approach. We use the *VGG16* model [1], a popular image classifier as our subject model and fifty randomly selected seed images from the ImageNet test dataset [4]. Our results suggest for 44 (out of 50) images, our approach can generate counterfactual explanations. Furthermore, in most cases, our approach can generate a counterfactual explanation by removing no more than two segments from the input image.

The remainder this paper is organized as follows. Section II provides an introduction to Deep Neural Network-based image classifiers, counterfactual explanations, and BENs. In Section III, we present our approach and give an example to illustrate the approach. Section IV reports the experimental evaluation of our approach, where we present our experimental design, results and discussion. Section V discusses the existing work on XAI. Section VI provides concluding remarks and directions for our future work.

II. Background

A. Deep Neural Networks

Deep learning is used across domains such as autonomous driving, speech recognition, speech translation, and medical imaging. At the core of deep learning is a Deep Neural Network (DNN) that is used to perform tasks such as image classification, object detection, and others. A DNN follows a neural network architecture and consists of an input layer, several hidden layers and an output layer. A trained DNN model takes an input (e.g., an image) and produces a prediction as output.

Compared to traditional software development, where the programming logic is implemented based on rules derived from the requirements, DNN based applications derive their decision logic (learning) from a training dataset. The decision logic is referred to as the trained DNN model.

In recent years, deep learning-based image recognition software systems have improved significantly and could be more efficient than humans in some domains. A practitioner can build a DNN model using different types of neural network architecture. One of the popular neural network architectures used for image recognition tasks is convolutional neural networks (CNN). Given an input, CNN architecture is known for its ability to detect important features without any human supervision. The subject models used in our experiments use a CNN based architecture and perform image classification.

B. AI Explanations

The explanations generated by XAI tools can be categorized into two types, feature-importance based explanations and counterfactual explanations. Assume a model M that produces a classification X for an input image I. A feature-importance based explanation identifies a set of important features of I that contribute to decision X. In addition, it assigns weights to the features that quantify their contribution. In contrast, a counterfactual explanation identifies a minimum set of features of I that if removed, shall change the prediction. In other words, counterfactual explanations are contrastive in nature.

C. BEN

Ghandehari et al. developed a combinatorial testing-based approach called BEN to software fault localization [7, 8]. Localizing a fault using BEN consists of two major phases: inducing combination identification (Phase I) and faulty statement localization (Phase II). BEN assumes that a combinatorial t-way test set is available and has been executed on the SUT. In the first phase, BEN takes the t-way test set and its results as input and tries to identify one or more inducing combinations in an iterative manner. BEN analyses the test file and identifies a set of t-way suspicious combinations. Based on the t-way suspicious combination(s), BEN generates a new t-way test set. For the new t-way test set, the user generates concrete tests, executes the tests, and records their execution status (either pass or fail). Then, the user provides the execution status back to BEN. This process is repeated until BEN identifies an inducing combination. Note that BEN expects the initial test set to contain at least one passing and one failing test. If there is no passing test in the initial t-way test set, BEN identifies an inducing combination based on the initial t-way test set. In our approach, the inducing combination identified by BEN is used to generate counterfactual explanations. Phase II of BEN is not utilized in our approach.

III. Approach

This section presents a combinatorial approach to generate counterfactual explanations for machine learning models that take an image as input and output a prediction. Our approach consists of four phases: Image segmentation, t-way testing, identifying inducing segments, and constructing explanations.

Image Segmentation: Image segmentation is a widely used image processing technique that partitions a digital image into different segments based on the characteristics of the image pixels. In our approach we first perform image segmentation on the input image. As discussed later, each segment is modeled as a parameter during CT. Working with segments instead of pixels allows us to reduce the number of parameters in our input parameter model (IPM).

We point out that the number of segments could potentially affect the quality of the counterfactual explanation. The more segments, the finer grained the resulting explanation could be. However, the more segments, the more parameters, the more expensive to produce the explanation. Many segmentation algorithms allow the user to define a maximum number of segments. The exact number of segments produced by the segmentation process is typically close to the maximum number. A trade-off decision often needs to be made when choosing the maximum number of segments.

Recall that BEN assumes that there exists an input parameter model (IPM) of the SUT, a test oracle to determine the status of the test execution, and a t-way combinatorial test set with execution results. In the following we discuss how to provide these components in the context of XAI.

T-Way Testing: We begin this phase by deriving an input parameter model for the SUT, i.e., for the input image. For an input image, every segment is considered as a parameter.

Our approach aims to identify a minimum number of segments that, if removed, would change the prediction. To remove a segment, we perform a masking operation on the particular segment. In our approach, a segment can either be masked or not masked. Therefore, in the IPM, for each

parameter, we identify the following two values – true (masked) and false (not masked).

Then, we generate an abstract t-way test set using ACTS, a combinatorial test generation tool [14]. We derive the concrete tests by applying masking to specific image segments (as per the test case) using image-processing python libraries [13, 25, 33, 34]. We execute the concrete tests (images) and determine their execution statuses.

Given an image, the DNN model produces a class label (prediction) as output. To determine the execution status of a test, we define the test oracle as follows: On executing the model with a test image, if the output (class label) matches that of the original image, we consider it to be a passing test. If the output does not match the output of the original image, we consider it to be a failing test.

Identifying Inducing Combinations: We begin this phase by providing an initial test file (as input) to BEN. The initial test file includes parameters and values, the test strength, the initial t-way test set, and the execution status of each test. In each iteration, analyzing the test file, BEN either generates an additional set of tests or terminates by identifying inducing combination(s). For additional tests generated by BEN, we derive concrete tests (t-way images), execute the model with the test images and update their execution statuses. Then, we provide the updated test results to BEN.

This process continues until one of the stopping conditions is satisfied: (1) an inducing combination is identified by BEN, or (2) the user decides to stop the process. In the latter case, the top-ranked suspicious combination is considered to be the inducing combination, and we proceed to the next phase.

Constructing Explanations: In this phase, we derive explanations based on the inducing combinations in an iterative manner.

Given the nature of the XAI problem, an inducing combination identified by BEN may not be directly used to produce a counterfactual explanation. Consider a scenario where an input image has 20 segments (i.e., 20 parameters, and each parameter has two values - TRUE, FALSE). BEN identifies the following two inducing combinations: (segment_1 = FALSE, segment_4 = FALSE), (segment_2 = TRUE, segment_4 = FALSE).

The first inducing combination suggests a test retaining segment_1 and segment_4 shall fail (change the prediction). Even though all the test images that contain these two segments have a different classification, this inducing combination cannot be used to produce a counterfactual explanation, since it does not suggest any segments to be removed.

The second inducing combination suggests to remove segment 2 (masked) while retaining segment_4 in order to produce a different classification. This combination can be used to produce a counterfactual explanation as discussed next.

In general, an inducing combination that suggests the removal of one or more segments can be used to produce counterfactual explanations.

We begin to construct a counterfactual explanation by selecting the top-ranked inducing combination, generating an image based on the inducing combination (modified image), executing the model with the image, and recording its execution status. Suppose the prediction of the modified image differs from the prediction of the original image (fail). In that case, the approach stops, and the modified image is shown as an explanation to the user.

Otherwise, if the prediction of the modified image is the same as the prediction of the original image state (pass), we select the next ranked inducing combination and repeat the process, i.e., generate an image based on the inducing combination, and execute and compare its prediction with the original prediction.

This process is continued until either of the two conditions is satisfied: (1) the prediction of a modified image generated based on inducing combination(s) differs from the prediction of the original image; or (2) all the modified images generated based on the inducing combination(s) match the original prediction. In the first case, the modified image is shown as an explanation to the user. In the second case, we derive an explanation as follows.

First, we analyze the test suite and identify a test that (1) contains the inducing combination, and (2) the prediction differs from the original prediction (i.e., a failing test). If there is more than one test that satisfies the two criteria, we select a test with the least number of masked segments. Recall that our objective is to identify a minimal number of segments that, if removed, shall change the prediction.

Next, in addition to the inducing combination, we mask the additional segments whose values are true in the test in an incremental manner (one segment at a time), starting with the segments closer to the segments in the inducing combination. This process is repeated until the prediction of the modified image differs from the original prediction. The modified image is shown as an explanation to the user. Note that masking additional segments from a failing test is likely to produce a counterfactual explanation, since its prediction differs from the original prediction.

Example: We illustrate our approach using an example. Consider the image in Figure 1. It is assumed that the DNN model is executed with the image and the prediction result (P) is available.

To derive a counterfactual explanation, we begin with image segmentation, which identified the possible number of segments for the subject image as 20 (Figure 2).

Next, we build an IPM with 20 parameters; each parameter has two values: {TRUE and FALSE}. Then, we generate a 2-way test set (12 tests) using ACTS [14]. We derive the concrete tests (test images), execute the model with concrete tests, record and compare their execution statuses (P`) with the original prediction (P). Based on the execution statuses, we have four passing tests (P = P`) and eight failing tests (P != P`). A test file is generated, and it contains the IPM, the strength of the t-way test set, the t-way test set, and its execution status.

Next we provide the test file as input to BEN. After a couple of iterations, BEN identifies an inducing combination - *segment_10=TRUE,segment_12=TRUE,segment_17=false*. Note that at each iteration, we repeat the process of deriving, executing, and updating the status of the additional tests.

To derive a counterfactual explanation, we generate a modified image based on the inducing combination -

segment_10=TRUE,segment_12=TRUE,segment_17=FALSE.

Original image

Prediction: **pomegranate**

FIGURE 1

Segmentation

Number of segments = **20**

FIGURE 2

Counterfactual explanation

Masked segments = 10, 12

Prediction: **hip**

FIGURE 3

Although the inducing combination consists of three segments, the modified image will have two (out of three) segments, namely *S_10, S_12* masked, while no changes being made to *S_17*, as its value is *FALSE, i.e., not to mask the segment.* Then, we execute the model with the modified image, and its output (prediction) is compared to the output of the original image. The prediction of the modified image differs from the original prediction.

At this point, the approach terminates, and the modified image (Figure 3) is shown as a counterfactual explanation to the user.

IV. EXPERIMENTS

In this section, first, we present the design of our experiments including the research question, the subject model and selection of seed images, segmentation and masking techniques, and the metrics used to measure the effectiveness of our approach. Second, we present and discuss our results. Third, we compare the results of our approach with SHAP, a popular state-of-the art XAI tool. Finally, the threats to validity are discussed. The source code, data and/or artifacts have been made available at [27, 28]

A. Research Questions

The major research question of our evaluation is the following:

- How effective is BEN in generating counterfactual explanations for DNN-based image classifiers?

B. Model

We evaluate our approach using an open-source, pre-trained model – VGG16 [1]. The model uses a convolutional neural network architecture consisting of 13 convolution layers and three dense layers. VGG16 is used in evaluating similar explainable AI tools [2].

C. Seed Images

The ImageNet dataset is an extensive collection of visual images. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is an annual competition for evaluating algorithms for object detection and image classification [4]. The VGG16 model, a runner-up at the ILSVRC 2014 challenge, is trained using the ImageNet dataset with over 14 million images with 1000 classes.

In our experiments, we use the ILSVRC2017 test dataset, the latest test dataset from ImageNet (ILSVRC2017). The test dataset consists of 5500 images [4, 5]. We randomly selected fifty seed images.

D. Segmentation

The Simple Linear Iterative Clustering (SLIC) algorithm is used to perform image segmentation [13]. Based on the maximum number of desired segments provided by the user, the SLIC algorithm clusters pixels based on their color similarity and proximity in the image plane and create segments. In our experiments, we set the maximum number of segments to 25. However, the exact number of possible segments varies for each seed image. This is because the SLIC algorithm generates segments based on certain properties of an image.

E. Masking of Segments

An image consists of an array of dots referred to as pixels. Pixels of a color image can have a value in the range of 0 to 255. The value of 0 represents a black pixel, and the value of 255 denotes a white pixel. In our experiments, we mask a segment by setting all its pixels to the value of 0.

F. Metrics

The effectiveness of our approach is measured in terms of the quality of the counterfactual explanations it produces. The quality of a counterfactual explanation could be measured in different ways [21][32]. Ultimately, a counterfactual explanation should make sense to a human subject. This is however subjective.

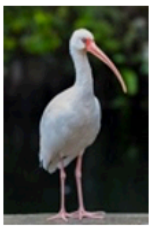
In our experiments, the quality of a counterfactual explanation is measured in the following two aspects: (1) the number of segments that need to be removed from the original image to produce the explanation. The fewer segments to be removed, the easier to be understood, the higher quality. (2) the explanation must produce a different prediction than the original prediction.

G. Results and Discussion

Our approach effectively derived counterfactual explanations for 44 (out of 50) seed images. In the following, we present the details of our results. Due to space limitations, we only show some example results in this section. The complete results are available at [27, 28].

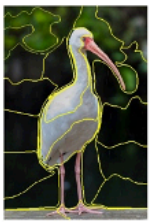
1) Counterfactual Explanations

First, we present the results of counterfactual explanations generated from an inducing combination alone i.e., no additional segments need to be removed. For 24 out of 50



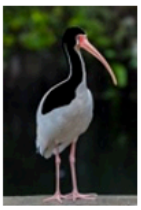
Original Image

Prediction: *white_stork*



Segmentation

Number of segments = **20**



Counterfactual Explanation

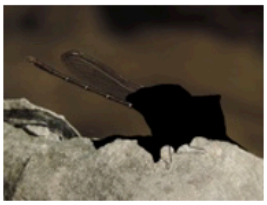
Prediction: *black_stork*

Original Image

Prediction: *dragonfly*

Segmentation

Number of segments = **23**



Counterfactual Explanation

Prediction: *lycaenid*

Original Image

Prediction: *stage*

Segmentation

Number of segments = **17**

Counterfactual Explanation

Prediction: *feather_boa*

Original Image

Prediction: *sea lion*

Segmentation

Number of segments = **20**

Counterfactual Explanation

Prediction: *promontory*

FIGURE 4 – COUNTERFACTUAL EXPLANATIONS DERIVED FROM INDUCING COMBINATIONS

images, the modified images generated based on their respective inducing combination (identified by BEN) effectively change the original prediction.

Our results show that for 6 out of 24 images, our approach removes one segment to produce the counterfactual explanation. For 16 out of 24 images, our approach only removes 2 segments. For the remaining 2 out of 24 images, our approach removes three segments.

Figure 4 shows some example results of these images. In each row, the first image is the original seed image; the second image shows the segmentation applied to the seed image. The third image is the counterfactual explanation. For the image in Row 1, removing one segment (segment 4) modifies the prediction from *white_stork* to *black_stork*. For the images in Row 2 (original prediction: *dragonfly*) and Row 3 (original prediction: *stage*), removing two segments changes the prediction to *lycaenid* and *feather_boa*, respectively. For the image in Row 4 removing 3 segments changes the prediction from *sea lion* to *promontory*.

Next, we discuss the counterfactual explanations that cannot be derived from the inducing combination alone. Instead, some additional segments need to be removed to produce a counterfactual explanation. For 20 images in our experiments, the modified images generated from their respective inducing combinations alone do not change the predicted class labels. Therefore, additional segments must be

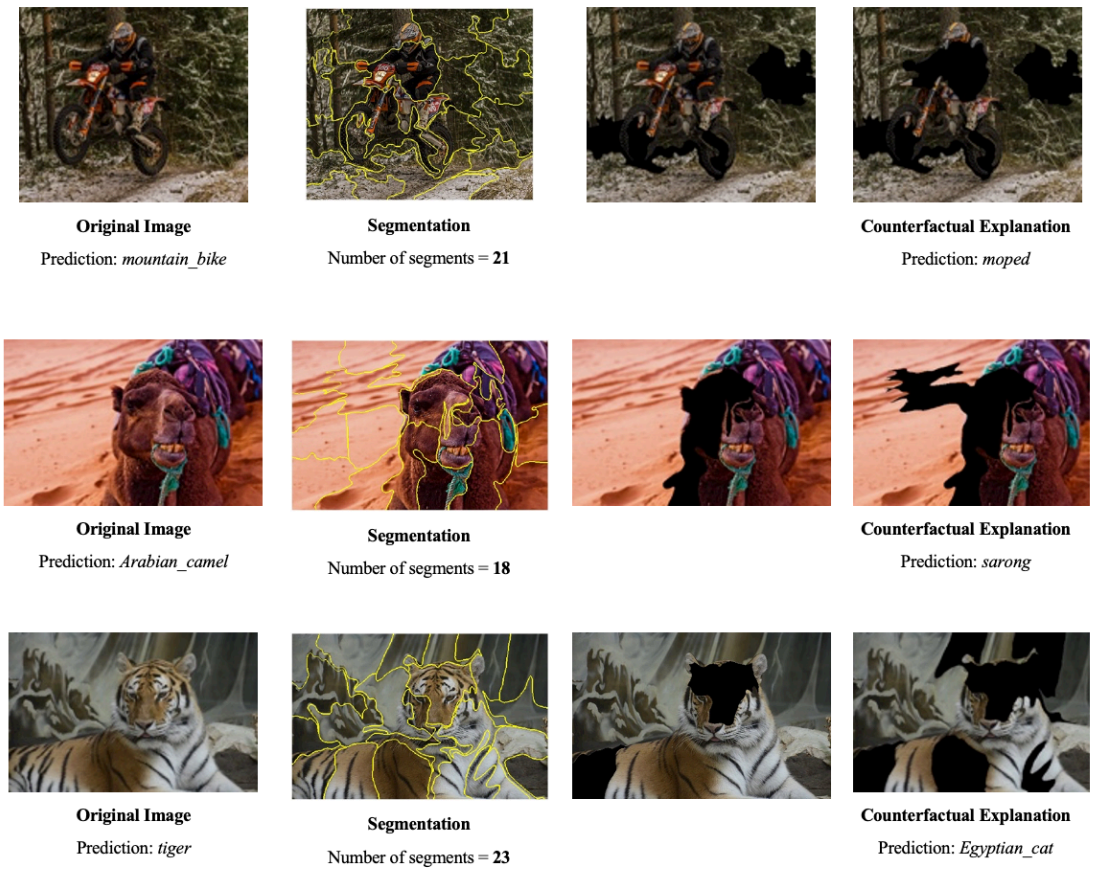


FIGURE 5 – COUNTERFACTUAL EXPLANATIONS DERIVED FROM INDUCING COMBINATION AND ADDITIONAL SEGMENTS

removed for these images in order to produce a counterfactual explanation.

Our results indicate that for 8 out of 20 images, masking one additional segment along with the inducing combination was sufficient to change the classification. For 7 out of the remaining 15 images, two additional segments needed to be masked. For the remaining 5 images, in addition to the inducing combinations, we masked three to five additional segments to generate a counterfactual explanation.

Figure 5 presents some of the counterfactual explanations generated from the inducting combination and one or more additional segments. In each row, the first image is the original seed image, followed by the segmentation applied to the seed image and the modified image produced based on their respective inducing combination. The fourth image is the counterfactual explanation produced from the inducing combination and one or more additional segments.

For the image in Row 1 - image #2737 with an original prediction - *mountain_bike*, masking one segment (segment_4=true), in addition to the inducing combination (segment_5=true, segment_13=true), changes the original prediction from *mountain_bike* to *moped.* Similarly, for image #4148 (Row 2) with an original prediction of *Arabian_camel*, masking one additional segment changes the original prediction from *Arabian_camel* to a *sarong.*

Consider the image in Row 3 (image #3793, original prediction – *tiger*), in addition to the inducing combination (segment_7=true, segment_19=true), masking four more segments (segment 2, 4, 17, 20) is necessary to change the original prediction from *tiger* to an *Egyptian_cat.*

The results suggest that in most cases, our approach can efficiently generate a high quality counterfactual for image classifiers. In other words, our approach can effectively identify a minimal (2 or 3 segments) yet important set of segments that if removed, would modify the original prediction.

We note that BEN was unable to identify inducing combinations for five seed images. For one of the seed images (image # 4541), BEN terminated with an error message. *There is no suspicious combination whose length is 2.* For the remaining five seed images, in spite of multiple iterations, BEN failed to identify an inducing combination. We observe that all the additional tests generated by BEN resulted in a passing status for each of these images. Therefore, we suspect BEN is unable to find an inducing combination as it expects at least one failing test to identify an inducing combination. We plan to investigate this as part of future work.

2) Comparison with SHAP

We compare the counter-factual explanations (derived by our approach) with SHAP, a widely used feature-importance approach tool [19]. Given an input and a pre-trained model, SHAP produces explanations for a model's decision by ranking the input features that contributed to the model's decision (feature-importance-based explanation). This comparison allows us to see the importance of the segments removed by our approach to produce a counterfactual explanation.

FIGURE 6 – COMPARISON WITH SHAP

Figure 6 presents some of the comparison results. The first image in each row presents the counterfactual explanation identified by our approach. The second image represents the output produced by the SHAP tool. SHAP output consists of four images: the original image (provided as input to the SHAP algorithm), followed by the top three predictions from the model with the features (segments) contributing to that corresponding predictions. Features (segments) that positively contribute to the outcome are highlighted in green, and features (segments) that negatively contribute to the outcome are highlighted in red.

Among the five images, the output from SHAP suggests, the set of segments that are removed to generate a counterfactual explanation in our approach positively contributes to the original decision (highlighted in green color). In other words, our approach identifies a minimal yet significant set of segments that if removed, shall modify the prediction. One of the interesting examples is the image from row 3 in Figure 6. The image consists of two performers, a microphone and a guitar. The predicted class label for the original seed image is a *stage*. Our approach suggests a part of a performer's body (segment #7) be removed to generate a counterfactual explanation, which is an unexpected segment (intuitively). However, the results from SHAP confirm that segment #7 contributes significantly to the predicted class label.

A counterfactual explanation does not determine the correctness of the original prediction. Consider the image from the image from row 5 in Figure 6. The image consists of a person on a motorcycle. The predicted class label for the original seed image (*mountain_bike)* might not match the user's expectation (*motorcycle)*. In such scenarios, i.e., in

case of a model's misprediction, deriving a counterfactual explanation can identify a set of features (segments) that if removed would modify the prediction. In other words, a counterfactual explanation could help identify a set of features that contribute to the misprediction. Likewise, SHAP also indicates the set of segments that contribute to the original prediction - *mountain_bike.* As part of future work, we plan to investigate the possibility of using the feedback from the counterfactual explanations to debug a model's misprediction.

This is an initial comparison study. The overall results indicate that BEN can be effectively adopted to derive a counterfactual explanation that indicates significant segments, i.e., segments that make a significant contribution to a model's decision. SHAP performs 1000 iterations to generate explanations. That is, SHAP executes the VGG16 model 1000 times to identify the important features that contribute to a model's decision. In contrast, our approach derives explanations with an average of 20 - 25 test cases (image perturbations). In other words, we derive a counterfactual explanation by executing the VGG16 model for an average of 25 times. We plan to perform a detailed, comprehensive comparison with other state-of-the-art explainable AI tools as part of future work.

H. Threats to Validity

Threats to internal validity are factors that may be responsible for the experimental results, without our knowledge. To mitigate the risk of human errors, we tried to automate as many tasks as possible, from generating synthetic images to executing the tests. Also, we have manually checked some of the results whenever any inconsistent or surprising results occur. For example, for image 3456 and 3462 all the initial tests resulted in a failure. In contrast, image 3703 and 3793 had a mix of passing and failing test. In such scenarios, we manually verified test results by inspecting the images and the prediction results.

Threats to external validity occur when the results from our experiments could not be generalized to other subjects. The DNN model architecture used in our study have been used in other studies [2, 3]. We randomly selected fifty seed images from ImageNet, a large, diverse dataset with more than 5000 images. This helps to alleviate the risk of lack of diverse images used in our study.

V. Related Work

In this section, we discuss existing work that is closely related to our work. First, we discuss existing work on counterfactual explanations. Dhurandhar et al. proposed a method that produces contrastive explanations. Their method identifies two sets of pixels: (1) A minimal set of features that are sufficient to obtain the current classification (pertinent positive); and (2) A minimal set of features that should be absent to obtain the current classification (pertinent negative) [10]. In contrast, we identify a minimal number of features (segments) that if removed (absent), will change the current classification.

Goyal et al. proposed a technique that generates counterfactual visual explanations [11]. Assume that for an input image I, model M predicts class A. Their approach generates a visual explanation that tries to answer the following question: How should the image I be different for the model to predict Class B instead of Class A? Our work is similar to theirs in terms of altering the input image and showing a modified image as a counterfactual explanation. However, in our approach, the modification is limited to removal of one or more segments from the original image, whereas they generate counterfactual explanation by identifying and replacing regions of the original image with regions from the image belonging to the counterfactual class.

Vermeire et al. proposed a model-agnostic approach to generate counterfactual explanations for image classifiers [3]. Our work is similar to theirs in terms of identifying segments that, if removed, shall change the classification. Our work is different from theirs in the following ways: They propose two methods, i.e., Search for Evidence Counterfactuals (SEDC) and Search for Evidence Couterfactuals with Target Counterfactual Class (SEDC-T). SEDC uses a best-first search approach to generate a counterfactual explanation. In SEDC-T, a counterfactual explanation is generated by removing segments (iteratively) to reach a predefined target class. In contrast, we use a combinatorial testing-based approach to generate counterfactual explanations, and our approach does not target a predefined class.

Hendricks et al. propose a method that produces a descriptive counterfactual text as an explanation to the end user [12]. Compared to this, our approach displays a modified image (with removed segments) to the end user.

Existing work reported in [16, 17] generates counterfactual explanations for tabular data. In contrast, our work generates counterfactual explanations for an image data.

Riberio et al. proposed LIME that generates local explanations based on input perturbations that probe a ML model and derive explanations [18,20]. Lundberg et al. proposed SHAP that generates explanations using game theoretic framework [19]. Similar to our work, LIME and SHAP create image perturbations by segmentation to derive an explanation. However, our work focuses on generating a counterfactual explanation, whereas their work focuses on identifying important features that contribute to the original decision.

Sun et al. proposed a statistical fault localization-based approach called DeepCover to generate explanations for image classifiers [2]. In their approach every pixel from the image is assigned a score in terms of their likelihood to contribute to the original decision. An explanation is derived by adding sufficient pixels (a subset of the original pixels) that shall produce the original decision. Our work is similar to their work in terms of using a software fault localization-based approach to derive explanations. However, our work differs in the following two ways: 1) we generate t-way test inputs (image perturbations) whereas their approach randomly selects and masks a set of pixels; and 2) we generate a counterfactual explanation whereas their explanation focuses on the pixels that contribute to the original decision.

Similar to our work, Kuhn et al. adopted a combinatorial fault location process and reported an approach that identifies a unique t-way combination that contributes to a model's decision [30][31]. Their approach is designed for tabular data. In contrast, our approach focuses on image-based classifiers and produces counterfactual explanations.

VI. Conclusion and Future work

In this paper, we present a combinatorial testing-based approach to explaining image classifiers. Our approach is model-agnostic, as it treats the underlying model as a black

box. We evaluated our approach using the VGG16 model [1] and seed images from the ImageNet dataset [4]. Our results suggest that for 44 (out of 50) images, our approach can effectively generate a counterfactual explanation. For 28 images, the counterfactual explanation is generated by removing no more than 2 segments. Overall, the results indicate BEN, a combinatorial testing-based fault localization approach, has the potential to be effectively applied and derive explanations for ML models.

In some cases (6 out 50 images), we are unable to derive counterfactual explanations using our approach. Based on the initial analysis, we suspect that BEN is unable to find an inducing combination as it expects at least one failing test. Therefore, as part of future work, we plan to investigate this by increasing the initial test strength or increasing the segment size or both.

In addition, we plan to continue our work in the following directions. First, in our current approach, a counterfactual explanation cannot be derived from the inducing combination alone for some images. Also, some inducing combinations suggest no changes to be made, i.e., *not to mask any segment.* We plan to investigate how to generate more effective inducing combinations. Second, in the case of a model's misclassification, a counterfactual explanation could help identify a set of features that contribute to the misclassification. We plan to investigate how to use the feedback from the counterfactual explanations for model debugging. Third, we plan to extend this work to generate a counterfactual explanation for ML models trained with tabular data. Finally, we plan to include additional subject models and perform a detailed, comprehensive comparison with similar XAI tools.

ACKNOWLEDGMENT

This work is supported by research grant (70NANB18H207) from Information Technology Lab of National Standards and Technology (NIST).

Disclaimer: Certain software products are identified in this document. Such identification does not imply recommendation by the NIST, nor does it imply that the products identified are necessarily the best available for the purpose.

REFERENCES

[1] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[2] Sun, Y., Chockler, H., Huang, X., & Kroening, D. (2020, August). Explaining Image Classifiers using Statistical Fault Localization. In *European Conference on Computer Vision*(pp. 391-406). Springer, Cham.

[3] Vermeire, T., & Martens, D. (2020). Explainable Image Classification with Evidence Counterfactual. *arXiv preprint arXiv:2004.07511*.

[4] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, *115*(3), 211-252.

[5] ImageNet DET_test_dataset , http://image-net.org/image/ILSVRC2017/ILSVRC2017_DET_test_new.tar.gz, Accessed: 2021-01-15

[6] ImageNet VGG16 Model with Keras, https://slundberg.github.io/shap/notebooks/ImageNet%20VGG16%20Model%20with%20Keras.html, Accessed: 2021-01-17

[7] Ghandehari, L. S., Chandrasekaran, J., Lei, Y., Kacker, R., & Kuhn, D. R. (2015, April). BEN: A combinatorial testing-based fault localization tool. In *2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)* (pp. 1-4). IEEE.

[8] Ghandehari, L. S., Lei, Y., Kacker, R., Kuhn, D. R. R., Kung, D., & Xie, T. (2018). A combinatorial testing-based approach to fault localization. *IEEE Transactions on Software Engineering*.

[9] Wachter, S., Mittelstadt, B., & Russell, C. (2017). Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.*, *31*, 841.

[10] Dhurandhar, A., Chen, P. Y., Luss, R., Tu, C. C., Ting, P., Shanmugam, K., & Das, P. (2018). Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In *Advances in neural information processing systems* (pp. 592-603).

[11] Goyal, Y., Wu, Z., Ernst, J., Batra, D., Parikh, D., & Lee, S. (2019). Counterfactual visual explanations. *arXiv preprint arXiv:1904.07451*.

[12] Hendricks, L. A., Hu, R., Darrell, T., & Akata, Z. (2018). Generating counterfactual explanations with natural language. *arXiv preprint arXiv:1806.09809*.

[13] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., & Süsstrunk, S. (2012). SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, *34*(11), 2274-2282.

[14] Advanced Combinatorial Testing System (ACTS), https://csrc.nist.gov/projects/automated-combinatorial-testing-for-software , Accessed: 2021-01-20

[15] Su, J., Vargas, D. V., & Sakurai, K. (2019). One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, *23*(5), 828-841.

[16] Mothilal, R. K., Sharma, A., & Tan, C. (2020, January). Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*(pp. 607-617).

[17] Guidotti, R., Monreale, A., Ruggieri, S., Pedreschi, D., Turini, F., & Giannotti, F. (2018). Local rule-based explanations of black box decision systems. *arXiv preprint arXiv:1805.10820*.

[18] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*.

[19] Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. In *Advances in neural information processing systems* (pp. 4765-4774).

[20] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016, August). " 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* (pp. 1135-1144).

[21] Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., ... & Herrera, F. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, *58*, 82-115.

[22] Das, A., & Rad, P. (2020). Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*.

[23] Adadi, A., & Berrada, M. (2018). Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, *6*, 52138-52160.

[24] Guidotti, R., Monreale, A., Giannotti, F., Pedreschi, D., Ruggieri, S., & Turini, F. (2019). Factual and counterfactual explanations for black box decision making. *IEEE Intelligent Systems*, *34*(6), 14-23.

[25] 2015, Open Source Computer Vision Library, https://github.com/opencv/opencv, Accessed: 2021-01-20

[26] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in science & engineering*, *9*(3), 90-95.

[27] XAI-Tool, https://github.com/cjaganmohan/XAI-Tool, Accessed: 2021-01-20

[28] XAI-Tool-Results-Dropbox, https://tinyurl.com/y3sc6qoy, Accessed: 2021-01-20

[29] Hilton, D. J., & JOHN, L. M. (2007). The course of events: counterfactuals, causal sequences, and explanation. In *The psychology of counterfactual thinking* (pp. 56-72). Routledge.

[30] Kuhn, R., & Kacker, R. (2019). An application of combinatorial methods for explainability in artificial intelligence and machine learning (draft) (pp. 7-7). National Institute of Standards and Technology.

[31] Kuhn, D. R., Kacker, R. N., Lei, Y., & Simos, D. E. (2020, October). Combinatorial Methods for Explainable AI. In *2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)* (pp. 167-170). IEEE.

[32] Molnar, C. (2020). *Interpretable Machine Learning*. Lulu. Com

[33] Module:Segmentation – skimage v0.19.0 dev docs, https://scikit-image.org/docs/dev/api/skimage.segmentation.html#skimage.segmentation.slic, Accessed: 2021-02-22

[34] scikit-image/slic_superpixels, https://github.com/scikit-image/scikit-image/blob/main/skimage/segmentation/slic_superpixels.py, Accessed: 2021-02-22.