Implementing "Challenges" to Evaluate Explainability Methods

JACOB RAST, Biomedical Engineering Department, Carnegie Mellon University, USA
ABHISHEK VIJAYAKUMAR, Machine Learning Department, Carnegie Mellon University, USA
ANNA CAI, Machine Learning Department, Carnegie Mellon University, USA

CCS Concepts: • **Human-centered computing** → **Heuristic evaluations**; • **Computing methodologies** → *Machine learning*.

Additional Key Words and Phrases: explainability, tabular data, image data

ACM Reference Format:
Jacob Rast, Abhishek Vijayakumar, and Anna Cai. 2023. Implementing "Challenges" to Evaluate Explainability Methods. 1, 1 (May 2023), 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Many forms of machine learning interpretability tools are in everyday use, from inherently explainable models such as decision trees and InterpretML's Explainable Boosting Machine [9] to post-hoc explainability tools such as LIME [10] and SHAP [7]. These interpretability tools are in use by a wide variety of people, ranging from machine learning engineers to data scientists to end users.

We implement a package for the evaluation of interpretability tools in the context of usage by data scientists that works with both structured tabular data and image data.

Our package can be installed from PyPI [2]. It is open-source, with source code repository located on GitHub [1]. We also provide robust documentation. [3]

2 RELATED WORK

Our work extends upon the work of Kaur et al., which studies the use of interpretability tools by data scientists [6]. They find two main themes in the use of such tools: first, they are used to provide evidence of model validity or quality; and second, they are used to find flaws in models that stem from flaws in underlying data.

In the context of usage by data scientists, we can thus isolate the following desirable properties for an interpretability tool:

(1) Interpretations must be understandable; a user must understand displayed values and figures and must be able to correctly assign meaning to components of the interpretation.

Authors' addresses: Jacob Rast, Biomedical Engineering Department, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, Pennsylvania, USA, 15213, jrast@andrew.cmu.edu; Abhishek Vijayakumar, Machine Learning Department, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, Pennsylvania, USA, 15213, abhishev@cs.cmu.edu; Anna Cai, Machine Learning Department, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, Pennsylvania, USA, 15213, annacai@cs.cmu.edu.

XXXX-XXXX/2023/5-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

(2) Interpretations must be able to capture evidence of flaws; a user must understand how typical behavior is reflected in an interpretation. Then, if a flaw exists in underlying data, a user must be able to identify an atypicality in the interpretation resulting from this flaw.
(3) Interpretations must be able to distinguish flaws: an interpretation reflecting a flaw in underlying data must provide sufficient information for a user to identify what form the flaw takes.

Kaur et al. study such properties of interpretability tools via both a survey and a contextual inquiry.

In their analysis of understandability, they find that "despite being provided with standard tutorials, few of participants were able to accurately describe what the visualizations were showing" [6]. They further noted that some users took the existence of an interpretation as evidence for the validity of a model, even if the interpretation was nonsensical or incomprehensible to the user. While this can be partially addressed by creating better tutorials or simpler interpretatbility methods, this is in large part a problem with user incentives.

The contextual inquiry performed by Kaur et al. focuses on whether users could detect and identify systemic flaws in data using interpretability tools. They presented practicing data scientists with 6 different challenges with flaws corresponding to 6 themes of issues data scientists face in the course of their work. They found that "each participant only identified 2.5 issues on average" [6], though performance varied widely depending on the type of flaw. For example, only 1 of 11 users identified that multiple values of a categorical feature had the same meaning, while 10 out of 11 users successfully identified a distribution shift between training and test data.

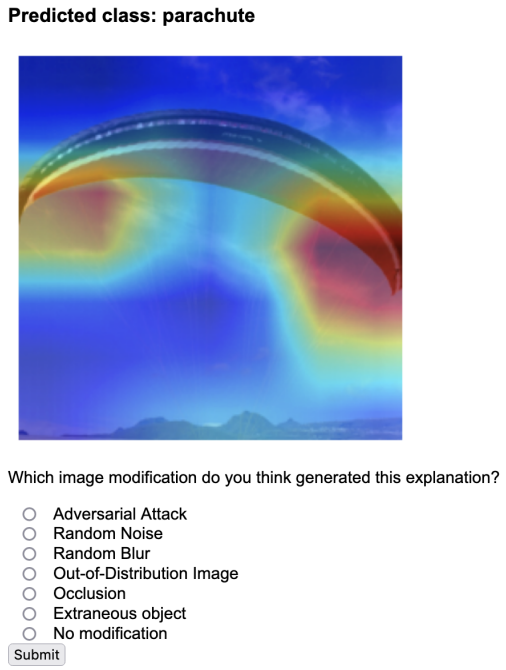
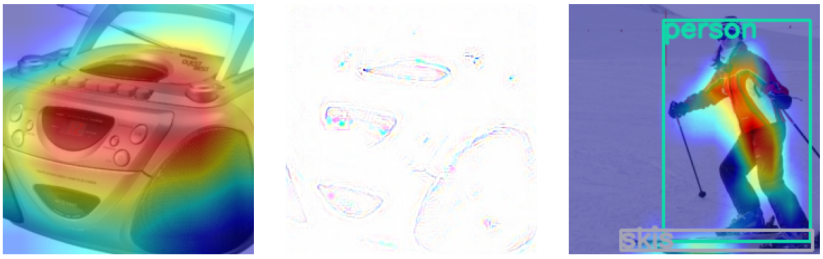


Fig. 1. Example of "explainability-challenges" UI in use



(a) A tape player with GradCAM heatmap (b) A tape player with Guided Backprop output (c) A skiier with EigenCAM heatmap

Fig. 2. Example of three image explanation techniques implemented in the ’explainability-challenges’ package

3 METHODOLOGY

Our package focuses on implementing challenges to explainability methods that replicate and extend the work done by the study, allowing evaluation of interpretability tools by how well users can detect the flaws in the challenges using them.

3.1 Notation

We define several interfaces for working with explainability methods and supervised learning tasks. Suppose we are given a supervised learning task with a matrix X of features for each sample and a vector y of labels. Further suppose this data is split into training and testing data: $X = \begin{bmatrix} X_{train} \\ X_{test} \end{bmatrix}, y = \begin{bmatrix} y_{train} \\ y_{test} \end{bmatrix}$.

We define a *manipulator* F as any function that modifies a dataset to introduce a flaw. That is, $F(X, y) = (X', y')$, where (X', y') is a dataset similar to (X, y) with a flaw introduced. Note that this can involve permuting the rows of the dataset; the train-test split need not be fixed.

We define a *challenge* as a triple (X, y, m), where (X, y) is a dataset and $m \in M$ is a model trained on (X, y) from the set of models M with a given architecture. We define a *challenger* as a set of challenges C given the following form:

- There exists a “base” challenge $(X, y, m) \in C$, for some $m \in M$.
- All other challenges (X', y', m') satisfy $(X', y') = F(X, y)$ for some manipulator F.
- All other challenges (X', y', m') satisfy $m' \in M$.

Intuitively, we start by training a model m on some dataset (X, y). We then create many variants of the dataset with flaws introduced, then train a model of the same architecture on each of these variants. The challenger thus consists of a suite of challenges that should only differ due to introduced flaws in the underlying data, so comparisons of explanations between challenges should focus only on differences in the underlying data.

We define an *explainer* E as a function mapping challenges to explanations. We intentionally do not specify the form these explanations may take, as different explainability techniques produce many different forms of explanation.

3.2 Structured Modality

3.2.1 Structured Manipulations. Our “explainability-challenges” package implements 5 manipulations for tabular data to simulate flaws described by Kaur et al.[6]

Fig. 3. Result of a feature explainer applying PartialDependence from InterpretML on a challenger

- Feature duplication (corresponding to Kaur et al.'s "redundant features"): an entire column is duplicated.
- Random replacement (corresponding to "missing values"): a proportion of samples is randomly selected and a column's values replaced with a user specified value.
- Ad-hoc categorization: a numeric column is binned to become categorical.
- Value split (corresponding to "duplicate data"): a proportion of samples with a user specified value of a categorical column is randomly selected and the value renamed.
- Distribution shift (corresponding to "changes in data"): the data is sorted by a user specified column, which is then respected by the train-test split.

The package also includes a wrapper for generating challenges and training scikit-learn models for each.

3.2.2 Structured Explanations. The package implements wrappers for applying global explainability methods to challengers, the "explainer" for methods that apply to all features and the "feature explainer" for method that apply to an individual (user specified) feature. We include samples of both using explainability methods from InterpretML (see Figure 3 for an example) in the package.

3.3 Image Modality

3.3.1 Image Manipulations. The "explainability-challenges" package implements several image manipulations commonly found in real-world data and known to cause difficulty for image classification models. These manipulations are as follows:

- Random noise. Gaussian noise with a user-specified standard deviation.
- Random blur. Gaussian blur with a user-specified kernel.

- Random occlusion. Occlude a random section of the image with a black box.
- Random dual-class. Occlude a random section of the image with an image drawn from a given dataset. This manipulation is intended to mimic the effect of having two valid classes in a target image.
- Adversarial manipulation. Adds structured random noise frequently invisible to the end user but which will change the model’s final classification decision. “explainability-challenges” implements the attack found in [5].
- Out-of-distribution manipulation. Simply return an image drawn from a distribution other than that upon which the model was trained.

Examples of these image manipulations can be found in figure 4.

3.3.2 Image Explanations. The “explainability-challenges” package also contains implementations of several methods used to explain image classification model behavior.

- Guided Backpropagation [12]. This method performs a more sparse form of backpropagation than that used to train the model, using both gradient and output information to backpropagate through nonlinearities. Performing guided backpropagation from output to input allows for visualization of the most important input features or pixels.
- GradCAM [11]. GradCAM will perform backpropagation to the last convolutional layer. The gradients are then passed through a global pooling average to determine neuron importance. Finally, neuron importance is passed through a ReLU nonlinearity such that only positive contributions are visualized.
- EigenCAM [8]. EigenCAM eschews backpropgation. This method simply takes the first principle component of the product of output logits and the last convolutional layer weights. While this method is more heuristic, it has been observed to work well in practice.

3.4 User Interface

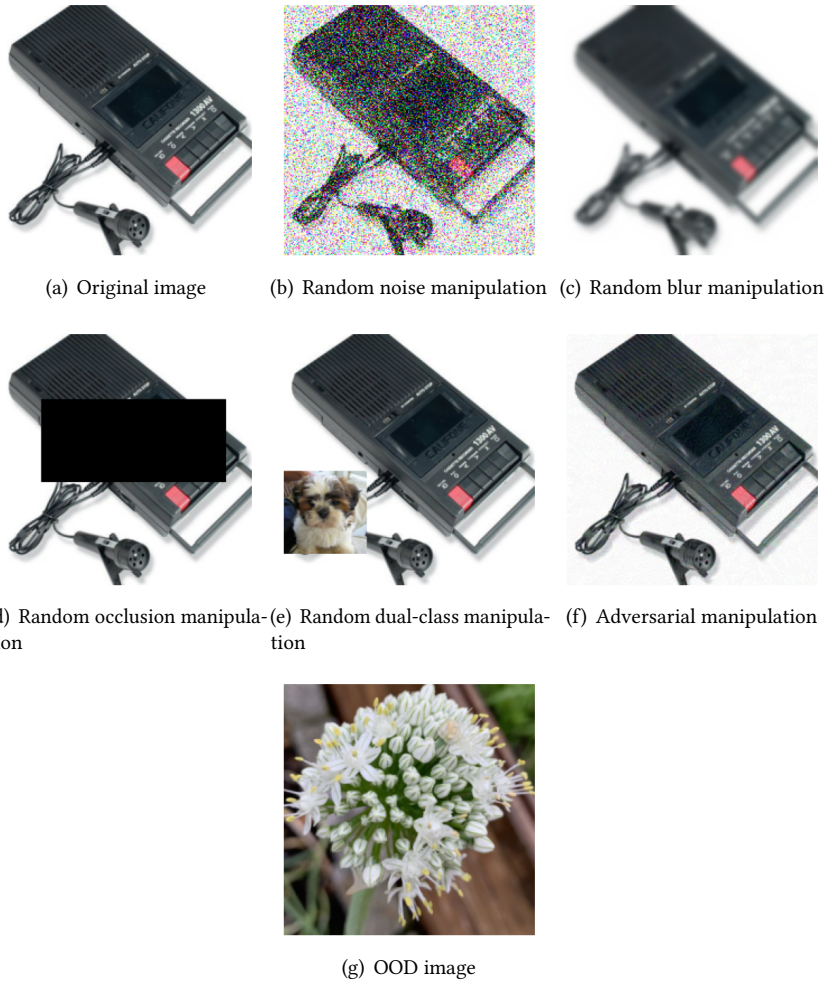
Finally, “explainability-challenges” includes a user interface that can be used to perform a user study similar to that proposed by Kaur et al. The interface will display an image and the model prediction for that image. A modification is randomly applied to the image and the explanation for that modified image is displayed.

From this information, the user will be asked to determine how the image was modified in order to produce the explanation shown. Note that the modified image will not be displayed. The accuracy of users’ ability to determine image modifications from explanations alone is used to evaluate the explainability method. An example of the UI in action can be seen in figure 1.

4 FINDINGS

The “explainability-challenges” package was used to run a preliminary user study. The purpose of the study was twofold: first, to evaluate the GradCAM image explanation method and second, to determine whether a user study implemented by explainability-challenges could yield a useful evaluation. Similar to the Kaur et al. study our user study focused on expert-level machine learning practitioners. Typically it is a challenge to find a user group with both qualifications in math statistics, a rigorous understanding of machine learning methods, and knowledge of explainability methods. Our study was able to identify a group of 8-10 machine learning practitioners to serve as users. The study administered three images drawn at random from the "ImageNet" database. The images were then modified using a random modification or no modification and passed to the “explainability-challenges” UI as described above.

All data used in the study can be found in figure 5. Users were presented with both the final row from figure 5 and the output predictions from a ResNet-18 model trained on the full ImageNet



(a) Original image (b) Random noise manipulation (c) Random blur manipulation

(d) Random occlusion manipulation (e) Random dual-class manipulation (f) Adversarial manipulation

(g) OOD image

Fig. 4. Example of all image modifications implemented in the 'explainability-challenges' package

dataset ("spotlight", "matchstick", and "parachute"). Note that images in rows 2 and 3 of figure 5 were not displayed such that ground truth information is not leaked to the users.

The panel of machine learning experts was asked to determine the image modification as a group given the information shown. Users predicted labels of "extraneous object", "adversarial attack", and "occlusion", whereas ground truth information was "blur", "adversarial attack", and "no modification". The method was then given a score of $\frac{1}{3}$ or 0.33.

Notable, there is strong justification for the labels the users gave. Image 1 shows a heat map localized in a single location and a mislabeled image, which could plausibly be interpreted as a hidden object occluding the image at the location shown. Image 3 shows a broken heatmap with two localizations at both ends. A plausible explanation for this heatmap is an occlusion blocking the center of the parachute.

These preliminary results give some indication that the benchmark task proposed in "explainability-challenges" is well suited for the evaluation of explainability methods. Users generate plausible

(a) Original image 1
Prediction: chainsaw

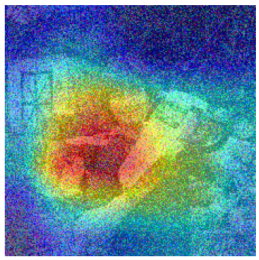
(b) Original image 2
Prediction: parachute

(c) Original image 3
Prediction: parachute

(d) Modified image 1
Modification: Random noise
Prediction: spotlight

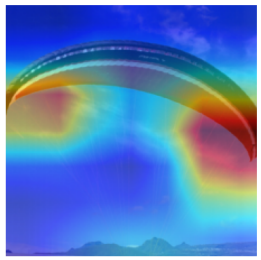
(e) Modified image 2
Modification: Adversarial attack
Prediction: matchstick

(f) Modified image 3
Modification: No modification
Prediction: parachute



(g) Explanation shown on modified image 1
Explanation: GradCAM

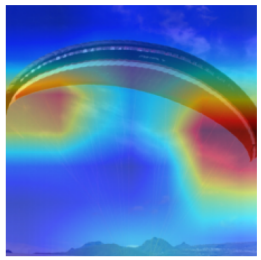
(h) Explanation shown on modified image 2
Explanation: GradCAM



(i) Explanation shown on modified image 3
Explanation: GradCAM

(j) Same explanation shown on original image 1
Explanation: GradCAM

(k) Same explanation shown on original image 2
Explanation: GradCAM



(l) Same explanation shown on original image 3
Explanation: GradCAM

Fig. 5. All images used in our user study

hypothesises from the explainability methods shown which do not match the corrupted data. In other words, the benchmark task was able to find scenarios in which explainability methods caused users to generate plausible but incorrect justifications of model outputs. Finally, this small-scale user study gave some indication of the shortcomings of GradCAM. Heatmaps lacked granularity and sufficient information to allow users to properly diagnose and identify data issues. However, our study was limited by size. No robust results can be generated using only three data points. Only preliminary indications can be gleaned, which may prove ultimately misleading given more data.

5 DISCUSSION

Our package can be used to perform user studies to evaluate the ability of interpretability methods to detect and identify flaws in underlying data. In this report, we summarize the features of the package and review a preliminary user study. Future work may entail comparing the results of evaluations by our package and other forms of evaluating interpretability methods. Additionally, more in-depth user studies may be conducted using either machine learning practitioners or Amazon mechanical turk users. We believe that our package could primarily serve as a tool to assist in the "Application-grounded Evaluation" and "Human-grounded Metrics" forms of evaluation suggested by [4].

REFERENCES

[1] 2023. explainability-challenges GitHub repository. https://github.com/inkyubeytor/explainability-challenges.
[2] 2023. PyPI explainability-challenges software package. https://pypi.org/project/explainability-challenges/.
[3] 2023. Read the Docs explainability-challenges documentation. https://explainability-challenges.readthedocs.io/en/latest/.
[4] Finale Doshi-Velez and Been Kim. 2017. Towards A Rigorous Science of Interpretable Machine Learning. arXiv:1702.08608 [stat.ML]
[5] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. arXiv:1412.6572 [stat.ML]
[6] Harmanpreet Kaur, Harsha Nori, Samuel Jenkins, Rich Caruana, Hanna Wallach, and Jennifer Wortman Vaughan. 2020. Interpreting interpretability: Understanding data scientists' use of interpretability tools for machine learning. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu HI USA). ACM, New York, NY, USA.
[7] Scott M Lundberg and Su-In Lee. 2017. 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., 4765–4774. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf
[8] Mohammed Bany Muhammad and Mohammed Yeasin. 2020. Eigen-CAM: Class Activation Map using Principal Components. In *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. https://doi.org/10.1109/ijcnn48605.2020.9206626
[9] Harsha Nori, Samuel Jenkins, Paul Koch, and Rich Caruana. 2019. InterpretML: A Unified Framework for Machine Learning Interpretability. *CoRR* abs/1909.09223 (2019). arXiv:1909.09223 http://arxiv.org/abs/1909.09223
[10] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "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, San Francisco, CA, USA, August 13-17, 2016*. 1135–1144.
[11] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2019. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *International Journal of Computer Vision* 128, 2 (oct 2019), 336–359. https://doi.org/10.1007/s11263-019-01228-7
[12] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. 2015. Striving for Simplicity: The All Convolutional Net. arXiv:1412.6806 [cs.LG]

Received 27 April 2023; revised 27 April 2023; accepted 27 April 2023