X-BaD: A Flexible Tool for Explanation-Based Bias Detection

Marco Pacini, Federico Nesti, Alessandro Biondi, Giorgio Buttazzo
Department of Excellence in Robotics & AI
Scuola Superiore Sant'Anna, Pisa, Italy
Email: {name}.{surname}@santannapisa.it

2021 IEEE International Conference on Cyber Security and Resilience (CSR) | 978-1-6654-0285-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/CSR51186.2021.9527956

***Abstract*—As widely known, machine learning has been thriving during the last two decades on the strength of two key factors: significant and continuous improvements in hardware performance and the possibility to produce large datasets through automated procedures. However, it has been shown that datasets often contain biases that can significantly affect the performance and resilience of machine learning models, e.g., when deployed to realize functionality for cyber-physical systems. For this reason, a lot of research has been devoted to methodologies and tools for detecting biases in the dataset.**

This paper presents X-BaD, a tool for bias detection designed to inject and discover biases in a neural network. It is implemented as an open-source Python library that extends the Spectral Relevance Analysis methodology. It allows data reusability and user customization by parameter configurations, and offers built-in functions to inject artificial biases into popular image datasets such as CIFAR-10, Pascal VOC, and ImageNet, for test purposes. This tool is compatible and extensible with features that are commonly used in machine learning frameworks, such as PyTorch and Pytorch Lightning datasets and models, Captum attributions, and Sci-kit Learn clustering algorithms and clustering performance evaluation methods. It also includes functions to interpret and assess the processed data. A set of experiments is finally presented to evaluate the effectiveness of the proposed tool.

I. Introduction

In recent years, machine learning algorithms achieved unprecedented performance in complex tasks such as computer vision [1], natural language processing [2], or game playing [3], [4], [5].

Thanks to these developments, machine learning models are being deployed on more and more cyber-physical systems, such as autonomous vehicles and advanced robots, originating new and different challenges in terms of resilience.

The success of machine learning is mainly due to the evolution of deep neural networks (DNNs) [6] trained on big datasets and executed on high-performance hardware accelerators, which helped boosting model performance and reduce computation times. Automated dataset augmentation techniques also helped increase the dataset size, further improving the performance.

However, datasets often include samples containing recurring patterns that may introduce a bias into the trained model. Such a bias can lead to wrong predictions. This phenomenon is sometimes referred to as Clever Hans behavior [7].

Identifying such behavior will lead to improvements in trustworthiness, resilience, and performance of cyber-physical systems. For instance, a bias in the perception system of an autonomous vehicle could lead to fatal consequences. Furthermore, systems based on biased DNNs are prone to security issues: an attacker could inappropriately interact with the system exploiting the bias. For instance, when a face recognition system is biased towards recognizing an individual using watermarks, the attacker could disguise himself only by adding a watermark to his photo.

Since detecting such a bias is quite hard for a human supervisor, a lot of work has been devoted to automatically detect such a bias using explainable AI (XAI) methodologies [8].

In particular, Lapuschkin et al. [9] have developed a methodology, called Spectral Relevance Analysis (SpRAy), to automate the discovery of *fixed-position bias* learned by a neural network. Here, fixed-position bias refers to a recurring pattern frequently found in the samples of a dataset at the same location (or, in general, on the same features). Figure 1 shows a few examples of a fixed-position bias (top images) and non-fixed-position bias (bottom images). SpRAy defines the following pipeline: first, a neural network is trained on a training set selected from the dataset; it then creates local explanations through some XAI method [8], selecting inputs from a test set extracted from the dataset; these explanations are post-processed by rescaling; finally, a clustering technique is used to process the newly obtained dataset. If more than one significant cluster is detected, SpRAy indicates the potential presence of fixed-position bias in the dataset.

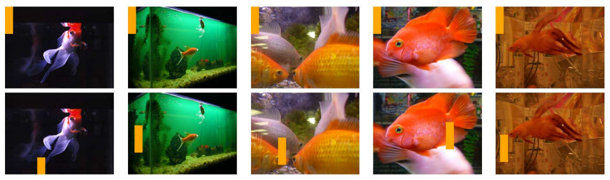

Fig. 1. **Top:** Sample images of the *goldfish* class from ImageNet augmented with fixed-position bias. **Bottom:** The same images augmented with non-fixed-position bias.

The original SpRAy implementation has shown compelling results, but it only considers one XAI method, namely Layer-wise Relevance Propagation (LRP) [10], and one clustering algorithm, namely spectral clustering [11]. However, it would be of great value to extend such a methodology to make it applicable to different datasets, models, explainers, and clustering methods. Furthermore, a more general methodology would allow investigating how dataset transformations affect training and explanations methods and it would enable the assessment of different explainers in their ability to detect bias.

For the reasons discussed above, this paper presents a tool, called eXplanation-based Bias Detector (X-BaD), which extends the SpRAy methodology to integrate the most popular Python ML frameworks, such as PyTorch [12] and PyTorch Lightning [13] for neural network training, Sci-kit Learn [14] for clustering techniques, and Captum [15] for neural network explanations. It is implemented as a light-weight Python library to be modular, easily extensible, and customizable. In fact, X-BaD can digest user-provided datasets and inject into them artificial bias, such as padding, watermarks, and custom patches. The neural network architecture, the explainer, the rescale factor, and the clustering algorithm can also be customized.

The tool also provides pre-constructed pipeline steps to improve experiment traceability and visualization. It enforces data reusability since, when repeating similar experiments, the tool fetches the useful data generated in the previous run, saving time and memory, which can be crucial while running multiple distinct pipelines with some shared parameters.

To summarize, this paper presents the following contributions:

- It presents a generalization of the SpRAy methodology to allow using different explainers, saliency maps post-processing, and clustering methods;
- It presents the functionalities of X-BaD, a Python tool designed to implement such a generalized methodology;
- It reports a set of experiments to validate the functionalities of the tool, by injecting artificial bias, and evaluating the detection capabilities of the methodology with different models, datasets, explainers, post-processing algorithms, and clustering methods.

The remainder of the paper is organized as follows: Section II presents the background and related work, Section III details the functionalities and implementation of the tool, Section IV provides experimental validation of those functionalities, Section V states the conclusions and future work.

II. Background and related Work

The trustworthiness and resilience of DNN-based systems mainly depend on the quality of the training dataset. Since training an efficient DNN requires a large number of samples, data collection is often automated, but automation is prone to introducing undesired and recurring patterns in the data. Training a DNN on those biased datasets may lead to the before-mentioned Clever Hans phenomenon [7], undermining the trustworthiness and resilience of the entire system.

To tackle this issue, Nadeem et al. [16] and Giloni et al. [17] proposed automated techniques to detect bias in scenarios dealing with text and tabular data. The literature on automated bias detection in computer vision scenarios is still limited: Tong et al. [18] use XAI methods to explore test data and search for undesired bias in image datasets; the same methodology is applied and automated by Lapuschkin et al. [9], and further extended in the presented paper.

XAI methods are heterogeneous, but the type of explanations provided can be categorized into two main categories: local and global explanations.

Local explanations focus on highlighting the features of the input (e.g., the pixels of an image or the words in a sentence) that are responsible for the corresponding output. Some of the most popular local explanations methods are Saliency maps [19], Occlusion [20], Integrated Gradients (IG) [21], and LIME [22]. Saliency maps are based on the computation of the gradient of the chosen output with respect to the input, and it was the first of a series of gradient-based explanations that were proposed in the literature. IG is based on the averaging of several gradients, computed with respect to perturbed versions of the original input. LIME (Local Interpretable Model-Agnostic Explanations) approximates the black-box model locally to the specific input that requires explanations. Many other explainers were presented in the literature and are not included in this section for space limitations. The interested reader can refer to Bodria et al. [8] for a detailed survey of XAI methods.

Conversely, global explanations try to visualize what internal neurons learned from the entire training set. For example, Mahendran et al. [23] proposed a method for reconstructing images from internal neural representations by searching for patterns that maximize the output for a given class. Bias discovery through global explanations is not considered in this paper and it will be part of a future work.

Both local and global explanations present limitations: global explanations rely on stochastic optimization methods and different initializations may bring completely different explanations; this means that finding bias is not always possible. On the other hand, local explanations are specific for a certain input. This makes a human-supervised search unfeasible for large datasets, such as ImageNet, but enables automation with algorithms.

To solve these problems, Lapuschkin et al. [9] proposed the SpRAy methodology, which processes a set of inputs through LRP [10] to produce explanations and then applies spectral clustering [11] to search for particular recurring patterns that form clusters of explanations.

Although SpRAy showed to be quite effective to detect biases, it only considers a single XAI method (LRP) and a single clustering algorithm (spectral clustering). Having a more general framework capable of handling different XAI and clustering methods would allow assessing them to improve the overall system performance.

For this reason, this paper presents a general tool that extends the SpRAy pipeline to consider different XAI methods and clustering algorithms.

III. Framework Description

This section presents the X-BaD framework, which extends the SpRAy pipeline by making each module fully customizable. Figure 2 illustrates the data flow diagram used in X-BaD and shows some examples of the results obtained at the output of specific stages.

First, an image dataset, such as CIFAR-10, Pascal VOC [24], or ImageNet [25], is loaded in the tool and can be augmented by adding artificial bias in the form of boxes, watermarks, or padding, as those shown in Figure 3. From now on, both the loaded and the artificially biased dataset will be referred to as the dataset.

At the second stage, a neural network architecture is selected by the user. The tool provides some preset options, such as AlexNet [1], VGG16 [19], ResNet [26], with appropriate optimizers, learning rate, and learning-rate schedulers. The network is then trained on the training set of the selected dataset using PyTorch Lightning [13], producing a model as an output.

At the third stage, a local explainer has to be selected to generate explanations (e.g., heatmaps for image datasets) for the predictions obtained on the corresponding test set. These explainers are implemented within the Captum framework [15], which is a collection of XAI methods, including Saliency maps [19], Occlusion [20], Integrated Gradients (IG) [21], LIME [22], Gradient SHAP [27], Guided GradCAM [28], Layer GradCAM, DeepLift [29], and Guided Backpropagation [30]. Please note that Layer Relevance Propagation (LRP [10]) is not currently available in Captum, and, at the time of writing, a stable implementation with PyTorch has not been publicly implemented yet.

The entire set of explanations obtained as output can optionally be post-processed through rescaling (through max-pooling) or principal component analysis (PCA) [31] to reduce the dimension of the explanation space.

Finally, at the last stage, the user has to select a clustering algorithm to process the explanations for possibly separating those containing a bias. Clustering algorithms are implemented in the Sci-kit Learn toolkit. Available algorithms are spectral clustering [11] or OPTICS [32]. Many different clustering algorithms are available in the toolkit, and X-BaD is flexible enough to include any other method.

The assessment of the clustering quality is performed through the Adjusted Rand Index (ARI) formula [33], which is already implemented in the Sci-kit Learn toolkit, and offers interpretability, bounded range[1], and requires no assumption on the cluster structure, in contrast to other assessment methods, such as the Rand Index [33]. On the other hand, it requires the true labels of the clusters, which, in the case of an artificial bias injected, are known. Other assessment score techniques based on ground truth, such as Mutual Information based scores [34], will be considered in a future implementation of the library.

Listing 1 shows an example of script running a full experiment in X-BaD. The library relies on the `XBad` core class, which provides methods for training the chosen network, creating explanations, and clustering the explanations; all these methods are wrapped and executed in the correct order by the `run` method. Finally, the user can visualize the results using the `tsne` [35] method, which provides an interpretable 2D visualization of the obtained clusters. For those cases in which an artificial bias is added by the user, ground truth clusters can be shown as well. By comparing the two scatterplots with some clustering similarity metric, the user is able to assess the efficiency of the entire pipeline, as shown in Figure 4.

Note that X-BaD is implemented to enforce data reuse, meaning that, each time it runs, it tries to fetch the reusable data generated in the previous runs. For instance, if the user wants to compare the capability of two different explainers to detect the same bias on the same model, in the second run the bias addition and the model training stages are not run again, but rather reused from the previous experiment. The library is available on the GitHub repository in the following URL: https://github.com/pacinigit/xbad.

```python
from sklearn.cluster import OPTICS
from xbad import XBad

pipeline = XBad(
    base_folder = 'folder',
    dataset = 'CIFAR10',
    bias_probability = 0.2,
    bias_method = 'box',
    biased_class = 6,
    model = 'alexnet',
    explainer = 'saliency',
    rescale = 2,
    pca = True,
    clustering_method = OPTICS(
        min_samples = 20,
        xi = 0.05,
        min_cluster_size = 0.05
    )
)

pipeline.run()
pipeline.tsne()
```

Listing 1. Code snippet: as soon as the object of type `XBad` is instantiated, the entire pipeline is run, then the output can be visualized using t-SNE.

IV. Experimental Results

This section reports a set of experiments carried out to validate the capabilities of the X-BaD framework. Please note that the experiments are not aimed at searching for the best combination of the algorithms used at the different stages of the pipeline, but rather at validating the main features of the tool.

The explainers considered in the tests include Saliency, IG and Gradient SHAP, which are those that have shown the best performance. Following the same principle, the clustering algorithm used in the final stage is Spectral Clustering.

Experimental results are summarized in Tables I & II, where the clustering performance score (ARI) is shown when using different explainers, different rescale kernels and with or without the application of PCA.

The first experiments were carried out on the CIFAR-10 dataset, which was augmented by injecting a box bias on 50% of randomly chosen elements in *deer* class. This allowed providing a simple visual benchmark to assess the performance

[1]The ARI takes values in the range (-1, 1), where 1 indicates perfect match between predicted and true labels, and -1 indicates big mismatch.

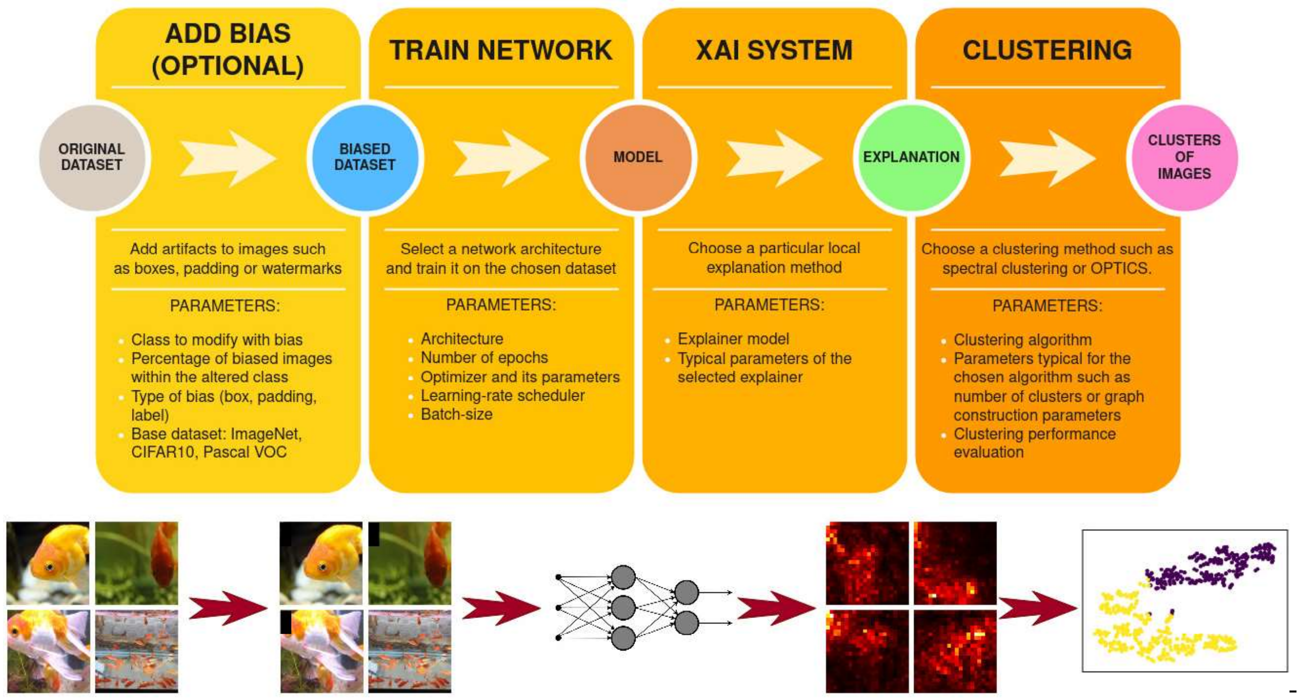
ADD BIAS (OPTIONAL)
TRAIN NETWORK
XAI SYSTEM
CLUSTERING
ORIGINAL DATASET
BIASED DATASET
MODEL
EXPLANATION
CLUSTERS OF IMAGES
Add artifacts to images such as boxes, padding or watermarks
PARAMETERS:
Class to modify with bias
Percentage of biased images within the altered class
Type of bias (box, padding, label)
Base dataset: ImageNet, CIFAR10, Pascal VOC
Select a network architecture and train it on the chosen dataset
PARAMETERS:
Architecture
Number of epochs
Optimizer and its parameters
Learning-rate scheduler
Batch-size
Choose a particular local explanation method
PARAMETERS:
Explainer model
Typical parameters of the selected explainer
Choose a clustering method such as spectral clustering or OPTICS.
PARAMETERS:
Clustering algorithm
Parameters typical for the chosen algorithm such as number of clusters or graph construction parameters
Clustering performance evaluation

Fig. 2. The X-BaD pipeline.

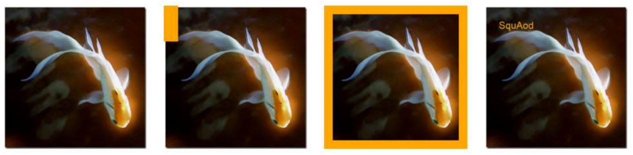

Fig. 3. A sample image of the *goldfish* class from ImageNet augmented using different artifacts: box, padding, and watermark, respectively.

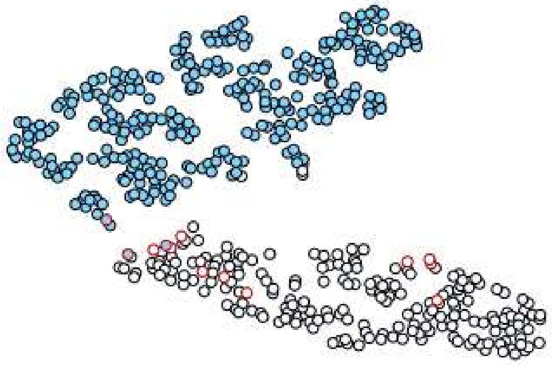

Fig. 4. t-SNE embedding produced by the X-BaD implementation shown in Table II performing IG without PCA. White and light blue points represent points belonging to the different clusters produced. Red-edged points are misclassified with respect to ground truth.

of the proposed tool. The neural network architecture selected was AlexNet, whose output layer was modified to accept the correct number of classes. It was trained for 6 epochs with an Adam optimizer with learning rate 0.001 and batch size 256. This setup resulted in about 76% of accuracy, which is lower with respect to the state-of-the-art result, but it can be acceptable for the purposes of this evaluation. All the experiments were performed on a single Tesla-V100 GPU of a DGX station.

TABLE I
ALEXNET ON AUGMENTED CIFAR-10: ARI & ARI WITH PCA

	Rescale				
	1	2	3	5	10
Saliency	0 0	0.95 0.95	0.96 0.95	0.96 0.96	0.96 0.96
IG	0.91 0.91	0.92 0.91	0.92 0.91	0.92 0.92	0.91 0.91
GradSHAP	0 0	0.02 0.02	0.03 0.06	0.03 0.08	0.11 0.08

Table I reports the quality of the clusters obtained from AlexNet under the setting discussed above, measured by the ARI metrics. Heatmaps have been generated from CIFAR-10 test set with same alterations used in the training set. Heatmap dimension was reduced by first applying rescaling (whose factor is reported in the table) and then applying PCA, where the number of selected principal components has been set to the minimum between the number of heatmaps (equal to the number of test samples of the *deer* class) and 80% of the rescaled heatmap features. The two numbers reported in each cell of the table correspond to the ARI using only rescaling and the ARI with rescaling followed by PCA, respectively.

As clear from the results, the explainer that showed the best performance, in this context, resulted to be the Saliency method, followed by IG, while Gradient SHAP did not produce significant results for automated bias detection. Also note that, when using the Saliency method, a rescale factor of at least two is necessary for detecting the bias automatically. This suggests that automatic bias detection procedures based on clustering evaluation need to execute an explainer multiple times, under different parameter configurations, and then integrate the results (e.g., using majority voting).

As a final remark, the low ARI values obtained using Gradient SHAP should not be interpreted as a poor performance of the explainer. In fact, from the heatmaps illustrated in Figure 5, it is clear that all the explainers used in this experiment were able to visualize the introduced bias and provide an meaningful explanation to humans. This suggests that the problem lies in the automated detection method based on clustering evaluation. Therefore, further investigation is required to make this technique suitable for different types of explainers.

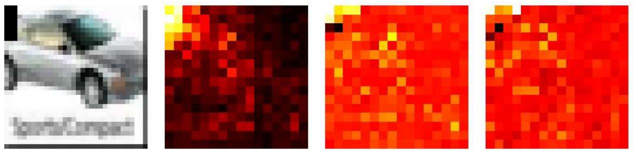

Fig. 5. **First image:** Input image from CIFAR-10 with box bias on the upper-left corner. **Second image:** Corresponding explanation of the output from a biased AlexNet, generated by Saliency and rescaled by a factor 2. **Third image:** Explanation obtained by IG in the same setting. **Fourth image:** Explanation obtained by Gradient SHAP in the same setting.

The second batch of experiments was carried out on the ImageNet dataset with a ResNet18 model. The training set was augmented with a box bias on 50% of randomly chosen elements in *goldfish* class. The network was trained from scratch for 70 epochs by stochastic gradient descent, learning rate 0.1, momentum 0.9, weight decay 0.0001, and batch size 256, using four NVIDIA A100 GPUs. The final model reached 72% Top-1 accuracy on the ImageNet test set augmented in the same way as the training set.

TABLE II
RESNET18 ON AUGMENTED IMAGENET: ARI & ARI WITH PCA

	Rescale						
	1	2	3	5	10	40	80
Saliency	0.29	0.06	0.03	0	0	0	0.14
	0.35	0.06	0.03	0.18	0	0	0.17
IG	0.77	0.87	0.92	0.91	0.89	0.84	0.81
	0.75	0.92	0.93	0.91	0.89	0.84	0.81
GradSHAP	0.34	0.50	0.45	0.44	0.37	0.32	0.29
	0.33	0.51	0.45	0.40	0.37	0.32	0.29

Table II reports the obtained results. Heatmaps have been generated from the first 500 images in the *goldfish* class of the training set; 60% of these images is augmented with box bias. As in the previous experiment, the heatmap dimension was reduced by first applying rescaling (whose factor is reported in the table) and then applying PCA, where the number of selected principal components has been set to the minimum between the number of heatmaps (equal to the number of test samples of the *goldfish* class) and 1% of the rescaled heatmap features.

In this test, carried out by training ResNet18 on the bias-augmented ImageNet, the explainer that exhibited the best performance was IG, using a rescale factor of three, whereas the others were not so effective in automatically detecting the bias.

The results obtained in the second test confirm the importance of running the detection procedure multiple times on different explainers and under different parameter configurations. This also justifies the availability of a flexible tool, like the one proposed in this paper, which can automate the detection procedure, allowing the user to easily configure and carry out a large number of experiments.

As noted in the previous test, the low ARI values obtained using Saliency and Gradient SHAP should not be interpreted as a poor performance of the explainers. In fact, from the heatmaps illustrated in Figure 6, all the explainers were able to visualize the bias in the the upper-left corner and provide an meaningful explanation to humans. This confirms that the problem lies in the high sensitivity of the clustering evaluation method to the specific explainer, data set, and, configuration parameters.

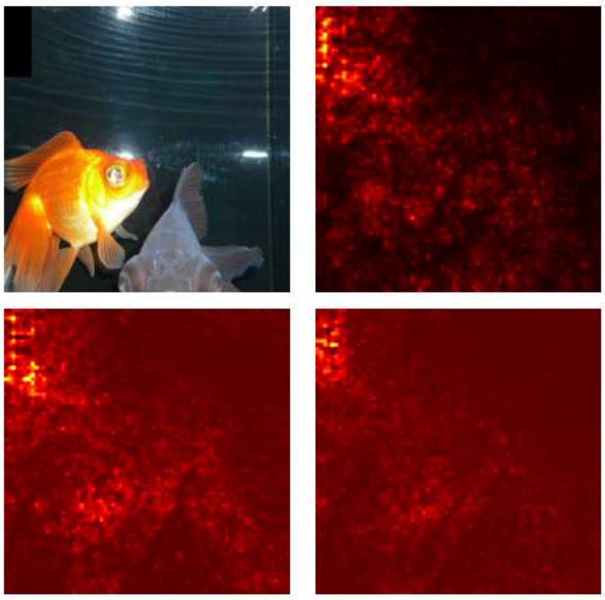

Fig. 6. **Top-left:** Input image from ImageNet with box bias on the upper-left corner. **Top-right:** Corresponding explanation of the output from a biased ResNet18, generated by Saliency and rescaled by a factor 3. **Bottom-left:** Explanation obtained by IG in the same setting. **Bottom-right:** Explanation obtained by Gradient SHAP in the same setting.

V. CONCLUSIONS

This paper presented X-BaD, a flexible tool for automating bias-detection procedures making use of explainers and clustering techniques. The tool implements and extends the SpRAy methodology [9] by allowing the user to inject different types of fixed-position biases in common datasets and then verify whether a specific neural network has learned such a bias.

A set of experiments has been carried out to validate the proposed tool on different neural networks, datasets, and configuration parameters. The obtained results consistently show that current automatic detection methods based on clustering evaluation are extremely sensitive to the specific explainer, dataset, and parameters used in the test (e.g., heatmap rescaling factor and number of principal components, if PCA is used for compressing the heatmap).

Such results suggest that, to increase the probability of automatically identifying a bias in the model, the detection procedure has to be executed multiple times, using different explainers and different parameter configurations, integrating all the obtained results by majority voting or other similar methods. Hence, the availability of a flexible tool that can automate the detection procedure, like the one proposed in this paper, represents a valuable support for users that intend setting up a large number of experiments.

The obtained results also show that, although all the used explainers provide an interpretable explanation where the bias is clearly visible to humans, only some of them produce correct results using the clustering evaluation technique. This suggests that further investigation on clustering methodologies is required to make them less sensitive to the different types of explainers.

As a future work, we plan to extend the tool for managing different types of XAI systems (including global explainers) and different types of dataset augmentation.

REFERENCES

[1] A. Krizhevsky et al., "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017. [Online]. Available: https://dl.acm.org/doi/10.1145/3065386

[2] T. B. Brown et al., "Language Models are Few-Shot Learners," *arXiv:2005.14165 [cs]*, Jul. 2020, arXiv: 2005.14165. [Online]. Available: http://arxiv.org/abs/2005.14165

[3] V. Firoiu et al., "Beating the World's Best at Super Smash Bros. with Deep Reinforcement Learning," *arXiv:1702.06230 [cs]*, May 2017, arXiv: 1702.06230. [Online]. Available: http://arxiv.org/abs/1702.06230

[4] V. Mnih et al., "Playing Atari with Deep Reinforcement Learning," *arXiv:1312.5602 [cs]*, Dec. 2013, arXiv: 1312.5602. [Online]. Available: http://arxiv.org/abs/1312.5602

[5] OpenAI and C. Berner et al., "Dota 2 with Large Scale Deep Reinforcement Learning," *arXiv:1912.06680 [cs, stat]*, Dec. 2019, arXiv: 1912.06680. [Online]. Available: http://arxiv.org/abs/1912.06680

[6] Y. LeCun et al., "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. [Online]. Available: http://www.nature.com/articles/nature14539

[7] D. Hothersall, *History of Psychology*. McGraw-Hill Education, 2004, google-Books-ID: HvgPAQAAIAAJ.

[8] F. Bodria et al., "Benchmarking and Survey of Explanation Methods for Black Box Models," *arXiv:2102.13076 [cs]*, Feb. 2021, arXiv: 2102.13076. [Online]. Available: http://arxiv.org/abs/2102.13076

[9] S. Lapuschkin et al., "Unmasking Clever Hans Predictors and Assessing What Machines Really Learn," *Nature Communications*, vol. 10, no. 1, p. 1096, Dec. 2019, arXiv: 1902.10178. [Online]. Available: http://arxiv.org/abs/1902.10178

[10] G. Montavon et al., "Layer-Wise Relevance Propagation: An Overview," Sep. 2019, pp. 193–209.

[11] U. von Luxburg, "A Tutorial on Spectral Clustering," *arXiv:0711.0189 [cs]*, Nov. 2007, arXiv: 0711.0189. [Online]. Available: http://arxiv.org/abs/0711.0189

[12] A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," p. 12.

[13] W. Falcon et al., "PyTorchLightning/pytorch-lightning: 0.7.6 release," May 2020. [Online]. Available: https://zenodo.org/record/3828935.YHhlHBKxVhF

[14] F. Pedregos et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, Jan. 2012.

[15] N. Kokhlikyan et al., "Captum: A unified and generic model interpretability library for PyTorch," *arXiv:2009.07896 [cs, stat]*, Sep. 2020, arXiv: 2009.07896. [Online]. Available: http://arxiv.org/abs/2009.07896

[16] M. Nadeem et al., "StereoSet: Measuring stereotypical bias in pretrained language models," *arXiv:2004.09456 [cs]*, Apr. 2020, arXiv: 2004.09456. [Online]. Available: http://arxiv.org/abs/2004.09456

[17] A. Giloni et al., "BENN: Bias Estimation Using Deep Neural Network," *arXiv:2012.12537 [cs]*, Dec. 2020, arXiv: 2012.12537. [Online]. Available: http://arxiv.org/abs/2012.12537

[18] S. Tong and L. Kagal, "Investigating Bias in Image Classification using Model Explanations," *arXiv:2012.05463 [cs]*, Dec. 2020, arXiv: 2012.05463. [Online]. Available: http://arxiv.org/abs/2012.05463

[19] K. Simonyan et al., "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps," *arXiv:1312.6034 [cs]*, Apr. 2014, arXiv: 1312.6034. [Online]. Available: http://arxiv.org/abs/1312.6034

[20] M. D. Zeiler et al., "Visualizing and Understanding Convolutional Networks," *arXiv:1311.2901 [cs]*, Nov. 2013, arXiv: 1311.2901. [Online]. Available: http://arxiv.org/abs/1311.2901

[21] M. Sundararajan et al., "Axiomatic Attribution for Deep Networks," *arXiv:1703.01365 [cs]*, Jun. 2017, arXiv: 1703.01365. [Online]. Available: http://arxiv.org/abs/1703.01365

[22] M. T. Ribeiro et al., ""Why Should I Trust You?": Explaining the Predictions of Any Classifier," *arXiv:1602.04938 [cs, stat]*, Aug. 2016, arXiv: 1602.04938. [Online]. Available: http://arxiv.org/abs/1602.04938

[23] A. Mahendran and A. Vedaldi, "Visualizing Deep Convolutional Neural Networks Using Natural Pre-images," *International Journal of Computer Vision*, vol. 120, no. 3, pp. 233–255, Dec. 2016. [Online]. Available: http://link.springer.com/10.1007/s11263-016-0911-8

[24] M. Everingham et al., "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010. [Online]. Available: https://doi.org/10.1007/s11263-009-0275-4

[25] J. Deng et al., "ImageNet: A large-scale hierarchical image database."

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 770–778. [Online]. Available: http://ieeexplore.ieee.org/document/7780459/

[27] S. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," *arXiv:1705.07874 [cs, stat]*, Nov. 2017, arXiv: 1705.07874. [Online]. Available: http://arxiv.org/abs/1705.07874

[28] R. R. Selvaraju et al., "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization," *International Journal of Computer Vision*, vol. 128, no. 2, pp. 336–359, Feb. 2020, arXiv: 1610.02391. [Online]. Available: http://arxiv.org/abs/1610.02391

[29] A. Shrikumar et al., "Learning Important Features Through Propagating Activation Differences," *arXiv:1704.02685 [cs]*, Oct. 2019, arXiv: 1704.02685. [Online]. Available: http://arxiv.org/abs/1704.02685

[30] J. T. Springenberg et al., "Striving for Simplicity: The All Convolutional Net," *arXiv:1412.6806 [cs]*, Apr. 2015, arXiv: 1412.6806. [Online]. Available: http://arxiv.org/abs/1412.6806

[31] M. E. Tipping and C. M. Bishop, "Mixtures of Probabilistic Principal Component Analysers," p. 30.

[32] M. Ankerst et al., "OPTICS: ordering points to identify the clustering structure," *ACM SIGMOD Record*, vol. 28, no. 2, pp. 49–60, Jun. 1999. [Online]. Available: https://doi.org/10.1145/304181.304187

[33] W. M. Rand, "Objective Criteria for the Evaluation of Clustering Methods," *Journal of the American Statistical Association*, 1971, publisher: [American Statistical Association, Taylor & Francis, Ltd.]. [Online]. Available: https://www.jstor.org/stable/2284239

[34] N. X. Vinh et al., "Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance," *The Journal of Machine Learning Research*, vol. 11, pp. 2837–2854, Dec. 2010.

[35] L. van der Maaten et al, "Viualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, Nov. 2008.