2022

ExplainabilityAudit: An Automated Evaluation of Local Explainability in Rooftop Image Classification

Duleep Rathgamage Don
Kennesaw State University, drathgam@students.kennesaw.edu

Jonathan Boardman
Kennesaw State University, jboardma@students.kennesaw.edu

Sudhashree Sayenju
Kennesaw State University, ssayenju@students.kennesaw.edu

Ramazan Aygun
Kennesaw State University, raygun@kennesaw.edu

Yifan Zhang
Kennesaw State University, yzhang60@kennesaw.edu

See next page for additional authors

Authors

Duleep Rathgamage Don, Jonathan Boardman, Sudhashree Sayenju, Ramazan Aygun, Yifan Zhang, Bill Franks, Sereres Johnston, George Lee, Dan Sullivan, and Girish Modgil

ExplainabilityAudit: An Automated Evaluation of Local Explainability in Rooftop Image Classification

Duleep Rathgamage Don[1], Jonathan Boardman[1], Sudhashree Sayenju[1], Ramazan Aygun[1], Yifan Zhang[1], Bill Franks[1], Sereres Johnston[2], George Lee[2], Dan Sullivan[2] and Girish Modgil[2]

[1]School of Data Science and Analytics, Kennesaw State University, USA
[2]The Travelers Companies, Inc., USA

***Abstract*—Explainable Artificial Intelligence (XAI) is a key concept in building trustworthy machine learning models. Local explainability methods seek to provide explanations for individual predictions. Usually, humans must check these explanations manually. When large numbers of predictions are being made, this approach does not scale. We address this deficiency for a rooftop classification problem specifically with *ExplainabilityAudit*, a method that automatically evaluates explanations generated by a local explainability toolkit and identifies rooftop images that require further auditing by a human expert. The proposed method utilizes explanations generated by the Local Interpretable Model-Agnostic Explanations (LIME) framework as the most important superpixels of each validation rooftop image during the prediction. Then a bag of image patches is extracted from the superpixels to determine their texture and evaluate the local explanations. Our results show that 95.7% of the cases to be audited are detected by the proposed system.**

***Index Terms*—explainability, computer vision, artificial intelligence**

I. Introduction

In the last decade, the explosion in deep learning technology has yielded machine learning systems capable of matching or surpassing human-level performance for some application domains such as object recognition [1]. Incredibly, these models appear to retain their superhuman performance on unseen test data, indicating that they are actually learning robust patterns. The introduction of additional layers or residuals from earlier layers continued to improve the performance [2]. As the complexity of models has increased, model interpretability has decreased. The inability to explain or comprehend such black box models is problematic in high-stakes problem domains, where safe and reliable performance are critical due to the high cost associated with errors [3].

XAI plays an important role in understanding black box models and the choice of method and nature of the explanation should be informed by the problem context. Many different approaches to interpretability have emerged to meet this demand, and they can be categorized along many dimensions – global vs. local, model-specific vs. model-agnostic, intrinsic vs. post-hoc [4], [5].

This paper presents an automated explainability audit technique named as *ExplainabilityAudit* to investigate the local interpretability in rooftop detection. As shown in Fig. 1, our method analyzes the reliability of classification by processing the explanations. This audit is a type of sanity check for the original classifier, primarily manifested as the technical perspective of explainability auditing [6]. ExplainabilityAudit analyzes the explanations and returns *satisfactory* if the explanations are good, or returns *weak* if the explanations are poor. This process requires training another model based on explanations as shown in Fig. 2. If this audit model determines that an explanation of a decision by the main model is *weak* (not reliable), this would require the involvement of a human expert to analyze the prediction and explanation. Human experts would only be required to step in for relatively few cases instead of potentially thousands or millions.

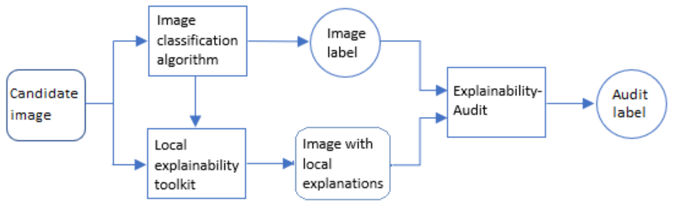

Candidate image
Image classification algorithm
Image label
Local explainability toolkit
Image with local explanations
Explainability-Audit
Audit label

Fig. 1. The image prediction-explainability pipeline: first the image recognition algorithm predicts the label of a candidate image. Then the local interpretability toolkit generates local explanations for each prediction. Finally, the ExplainabilityAudit algorithm works on both the image label and respective local explanations to decide if the prediction is reliable.

As an application, we propose a version of *ExplainabilityAudit*, which uses the original LIME framework as the local explainability toolkit. The goal of rooftop classification is to distinguish flat roofs among various other types of roofs in a nadir rooftop image dataset where the footprint of rooftops is often surrounded by various neighboring objects such as ground, trees, vehicles, driveways, etc. The regions other than the rooftop are considered to be the background. Our method uses LIME to split a candidate image into many superpixels (LIME prediction of regions that are utilized for classification) to create a synthetic dataset using random perturbations of the candidate image. Then a locally weighted interpretable linear model is trained on the new dataset of superpixels. Next, superpixels corresponding to the highest estimated coefficients are chosen to be the top local explanations. Our experiment is limited to the extraction of the largest segment of local explanations for validation images and the assumption that

The Travelers Indemnity Company

the prediction is *satisfactory* if the local explanations represent rooftops or similar regions. In this case, we ignore the disparity of rooftop labels.

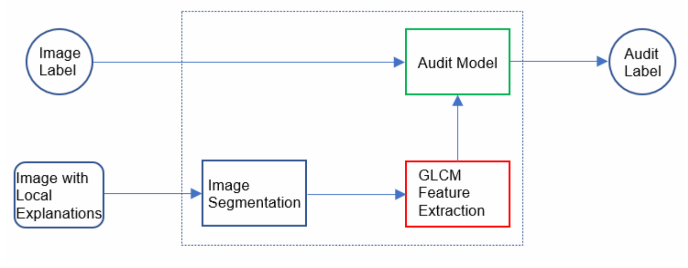


Fig. 2. The proposed ExplainabilityAudit method: The local explanations are segmented to extract GLCM features. The audit model evaluates the GLCM features with respect to the image label to predict the audit label.

This paper is organized as follows. The next section provides the related work on explainability and the background on local interpretability. Section 3 introduces our approach on auditing explainability for local interpretability. Experiments are presented and discussed in Section 4. The last section concludes our paper.

II. Related Work

Explainability generally falls into two main types of tasks: model understanding (global explainability) and decision understanding (local explainability). Model understanding or global explainability involves finding out how the model behaves for general data. Particularly, this means the task of recognizing the patterns in its predictive features or model parameters on classification. On the other hand, decision understanding or local explainability is concerned with the task of model behavior only on a particular data instance. Here, the aim is to find how the input features affect a single data point's classification. Most of the research in explainability focuses on decision understanding. Tools like what-if [7] offer dashboards called data point instance editor and feature statistics which help deduce explainability indirectly. Another popular method to generate explanations is using Shapley values [8]. The concept of Shapley values derives from game theory and is based on probability theory. Shapley values are the average marginal contribution of a feature across all possible coalitions. AWS SageMaker Clarify [9] uses Shapley values to give an explanation of a model.

The main reason why deep learning models are considered black boxes is that their behavior is not linear, and hence it is hard to come up with a global but simple interpretation. For decision understanding of a single data instance, it is not necessary to understand the complete non-linear behavior of the model. LIME [10] provides an explanation of the model around the local region of the data instances being scrutinized. These explanations have locally linear fidelity: linear behavior of the model in the vicinity of the prediction instance. LIME [10] learns the locally linear classifier by minimizing a loss function that minimizes the error between the actual model in the region and the explainable model. More comprehensive explainability tools like LIT [11], ELI5 [12] and Skater [13] include LIME to give model explainability.

Although the original LIME has some potential pitfalls, several important modifications are introduced in the past few years to address those issues. DLIME [14] is a deterministic version of LIME in which the random perturbation is replaced with agglomerative hierarchical clustering to create clusters within the training dataset and then applying K-nearest neighbor (KNN) to find the relevant cluster for the new observation. Then more stable explanations are generated by training a linear model over the selected cluster. ALIME [15] applies an alternative approach to reduce instability in generated explanations while maintaining local fidelity. In this method, a large number of synthetic data points are sampled from a Gaussian distribution and weighted for the locality by a denoising autoencoder. Lee et al. [16] state that the mean and the standard deviation of weighted superpixels of a test image produced by LIME demonstrate that the generated explanations are relatively stable. MPS-Lime [17] is a modified perturbated sampling for LIME that avoids correlation between the superpixels. In this method, the superpixels are represented by an undirected graph and the perturbed sampling is formalized as a clique-set construction problem. BayLIME [18] is a Bayesian extension to the LIME framework that applies prior knowledge and Bayesian reasoning to enhance the stability of the explanations.

III. Methodology

In this section, we first provide a brief description of LIME and then explain how we automate the evaluation of explainability in our research.

A. Local Interpretable Model-Agnostic Explanations (LIME)

Assume that the image recognition algorithm is a deep neural network represented by a real function f such that $f : X \mapsto Y, X \subseteq \mathbb{R}^d$, and $Y \subseteq \mathbb{R}$, where d is the number of RGB pixels of the image being predicted. For validation image $x \in X$, consider that $f(x)$ is the probability that x belongs to a certain class (e.g., 'flat roof' in our case). In this study, LIME splits x into d' number of superpixels such that each superpixel is a contiguous patch of similar pixels. An interpretable representation of x given by a binary vector x' is obtained such that $x' = \{0,1\}^{d'}$, where 0 and 1 represent the absence and presence of superpixels respectively. A random perturbation z is generated by blacking out some superpixels in x. Its interpretable representation z' is also a binary vector as above. Then a set of N number of random perturbations Z is generated by sampling uniformly at random around x' in order to train a locally weighted linear model $g \in G$, where G is the class of potential interpretable models. Also, the perturbations in the original presentation are recovered and predicted by using the image recognition algorithm f to obtain a target set $f(z)$ for the predicted class of x. A weight function Π_x is an exponential kernel defined on some distance function D, introduced as a proximity measure such that $\Pi_x(z) = exp\left(-D(x,z)^2/\sigma^2\right)$, where $D(x,z)$ is the distance between x and z, and σ is the kernel width [10].

To measure how unfaithful the local linear model g to the image recognition algorithm f, a loss function $\mathcal{L}$ is used such that

$$\mathcal{L}\left(f, g, \Pi_x\right) = \sum_{z,z' \in Z} \Pi_x\left(z\right)\left(f(z) - g\left(z'\right)\right)^2 \qquad (1)$$

Note that every linear model g is not simple enough to be interpretable. Therefore, another loss function Ω known as the measure of complexity is introduced to determine how complicated a linear model g is. $\Omega(g)$ could be the number of nonzero weights for a linear model. It is added to the loss in equation (1) to obtain the total loss. Finally, a linear model ξ is trained on the dataset consisting of z' and $f(z)$, and the local explanations are obtained from the linear model that minimizes the total loss.

$$\xi(x) = argmin_{g \in G}\, \mathcal{L}\left(f, g, \Pi_x\right) + \Omega(g) \qquad (2)$$

Setting ξ to be linear regression, the top local explanations are extracted as superpixels that represent the highest k estimated regression coefficients.

B. Preparing Local Explainability for Auditing

The local explanations generated in classifying image x by the image recognition algorithm f can be visualized as a new image denoted by ℓ. An *audit model* is built to classify the image ℓ by utilizing the class of x in order to evaluate the local explanations generated. The purpose of the audit model is to determine whether f has focused on the proper regions in the image or not.

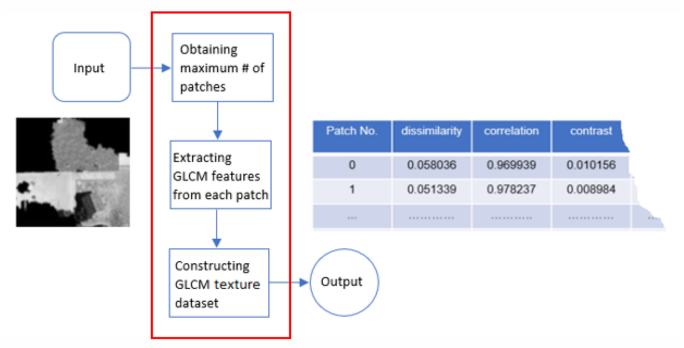
Although LIME shows some instability in the generated explanations, a typical image ℓ can be considered as a sparse representation of the corresponding x. Although ℓ and x have the same dimensions, it is possible that a very high percentage of pixels could be blacked out in ℓ. Large areas of black pixels aggravate the auditing task. Hence, the image ℓ cannot be directly used for auditing purposes.

C. Patch-based Preprocessing for Auditing

Patch-based preprocessing assumes that superpixels could be large enough and also, not every segment in a superpixel could be relevant for the correct explanation. Therefore, the preprocessing in Fig. 2 is applied to each image ℓ for patch-based processing. The stages are briefly explained in the following paragraphs.

Selecting Superpixels for Patch-based Analysis. First, the image ℓ is converted to an 8-bit greyscale image of which each pixel is represented by an integer $0 - 255$. Then appropriate masking is applied to the greyscale image and the extreme outer contour for each segment of the local explanations is determined. Note that these image segments might contain several neighboring superpixels. For our experiment, the image segment with the largest contour is selected and a rectangular bounding box is applied to the selected image segment to crop the greyscale image.

Grid Search for Maximizing Number of Patches. Ideally, the above preprocessing stage can be repeated for all such image segments that contain at least a single $p \times p$ pixel image patch. The selected superpixel has typically a non-rectangular convex or concave shape having many black regions close to the sides of the minimum bounding rectangle. The superpixel image can be divided into a grid containing $p \times p$ cells. However, because of the black regions and irregular shape of the superpixels, the patch selection process can be improved by not starting from the top left corner of the minimum bounding rectangle. A patch can be used for analysis if it has at least θ portion (patch coverage threshold) of its pixels as non-black. This problem can be solved using a grid search approach that maximizes the number of patches that satisfy the patch coverage threshold. A virtual grid of $p \times p$ pixel cells is created on the rectangular bounding box of the selected image segment. This virtual grid is initially aligned with the top and the left margin of the bounding box. Let the coordinates of the top left corner be (m, n), where m, n are integers such that $0 \leq m \leq p$, and $0 \leq n \leq p$. The number of image patches satisfying the patch coverage constraint is calculated for each value of m and n. The best m and n are chosen such that the number of image patches that lay in the grid is maximized and used to determine the bag of image patches. In this work, we use patch size as 10×10 considering the varying sizes of superpixels.


Input
Obtaining maximum # of patches
Extracting GLCM features from each patch
Constructing GLCM texture dataset
Output

Patch No.	dissimilarity	correlation	contrast
0	0.058036	0.969939	0.010156
1	0.051339	0.978237	0.008984
...	....	....	....

Fig. 3. The GLCM feature extraction: the input grayscale image is transformed into a bag of image patches. Then some GLCM features are extracted from each image patch to construct the GLCM dataset. (Satellite images courtesy of Nearmap US, Inc.)

Creating GLCM texture datasets. Before creating the bag of image patches, it is recommended that the selected greyscale image segments are converted into a lower bit depth such as 4-bit or 5-bit. This is an important preprocessing to effectively extract some Grey Level Co-occurrence Matrix (GLCM) features from each valid image patch to construct a GLCM texture dataset for the respective bag of image patches as shown in Fig. 3. If an 8-bit image patch was used for the above purpose, its GLCM would be a sparse matrix and some Haralick features might decrease in amplitude and generate a poor representation of the texture of the image patch [19]. In this GLCM texture feature dataset, the extracted GLCM features become columns, while image patches become rows.

D. *Patch-based Auditing*

The major motivation behind patch-based auditing is that a superpixel may cover proper regions as well as incorrect regions for decision-making. In the first stage of the audit model, a supervised machine learning algorithm is used to classify each patch available in the GLCM texture dataset into a set of classes formed by image labels and background. In the second stage, the entire explanation is classified based on the majority of voting. The next step is to compare this result with the image label. If the result matches the image label then the explanation is considered satisfactory or weak otherwise. In the output, the satisfactory class indicates that the segment is a proper region to make a decision whereas the weak class denotes that it may not be possible to make a correct decision or the decision could be unreliable.

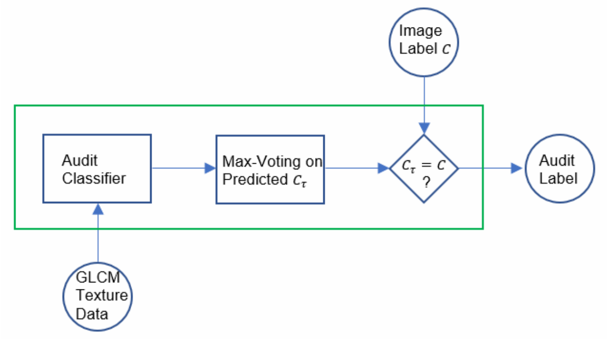


Fig. 4. The proposed audit model: The audit model utilizes a multimodal architecture in which an audit classifier applies the max-voting strategy on the prediction of patches and the result C_τ is compared with the predicted image label C. If both are equal, then the audit label is satisfactory and weak otherwise.

IV. Experiments and Evaluation

In this section, we explain the rooftop dataset used in the experiments, the tuning of LIME algorithm, and the classifiers used for auditing explanations. Then we provide the results of our experiments before the discussion. To conduct experiments, we have used AWS/Amazon SageMaker instance p3.2xlarge. The machine learning models were trained using TensorFlow and Keras 2 on Python 3 with CUDA 9.0 and MKL-DNN.

A. *Experimental Setup*

1) Datasets: The original dataset is provided by The Travelers Indemnity Company. This dataset is a nadir rooftop imagery consisting of 3715 RGB images split into 2956 training images and 759 validation images. The images have a fixed size of 640 $\times$ 640 pixels. The label sets have two nearly balanced classes: flat and non-flat. Each rooftop image is preprocessed to produce a polygonal bounding box consisting of the footprint of the rooftop and possible background objects. This dataset is used to train our image recognition (rooftop detection) model.

To extract patches, a dataset is created by randomly selecting and processing some 200 training images and 88 validation images belonging to the original dataset. The training images of this dataset are manually obtained by selecting rectangular regions of either rooftops or background but not both.

2) GLCM feature extraction: When extracting image patches, the patch coverage threshold used is θ = 0.95. The training GLCM texture dataset consists of 7162 observations, each representing a training patch belonging to one of some 200 training images and 6 GLCM features known as Dissimilarity, Correlation, Contrast, Homogeneity, Energy, and Angular Second Moment (ASM) obtained with angle 0 and distance 2. The observations are nearly balanced in terms of the class labels 'rooftop' and 'background'. Random sampling is used to create a smaller training GLCM texture dataset with 0.7 observations, and the rest is used for testing the audit classifier. Note that our experiments are limited to the largest segment of the local explanations in each image ℓ.

3) Tuning LIME algorithm: The following parameter settings are used for the LIME algorithm. The number of superpixels d', generated for each image x is in the range (80, 120). The number of random perturbations N is set to be 1000. The weighted linear g is linear regression. In the weight function Π_x, distance function D and the kernel width σ are cosine and 0.25 respectively. Setting k=8 led to the extraction of the best local explanations.

4) Rooftop Recognition: The rooftop detection algorithm is a deep neural network created by modifying a pretrained ResNet50 architecture using transfer learning. In this process, the top layer of the ResNet50 is replaced by three fully connected layers including a new top layer for binary classification. Then only the newly added layers are trained with the rooftop training dataset. After training on the complete training dataset, the performance of the image classification algorithm is validated using the complete test dataset, and the following performance measures are obtained: Accuracy: 0.8326, Precision: 0.7904, Recall: 0.8225, F-1: 0.8109, and ROC: 0.9159.

5) Audit Classifier: For the patch-based audit classifier, three binary classification algorithms known as Support Vector Machine (SVM), Artificial Neural Network (ANN) with one hidden layer and 10 neurons per layer, and K-Nearest Neighbor (KNN) are used. These algorithms are trained on the same training GLCM dataset using the following hyperparameters. For SVM with RBF kernel, gamma and cost are chosen to be 1 and 1000 respectively. For ANN built using the Sklearn MLPC module, the default batch size, optimization, and learning rates are used. The number of epochs is set to be 1000. For KNN, K is selected to be 15.

B. *Results*

Since the difference between image labels is ignored in this experiment, the local explanation is considered *satisfactory* if the majority of patches belong to a rooftop. Also, the local explanation is considered *weak* if the majority of patches belong to the background. We experiment with three different machine learning methods applying for the audit classifier: SVM, KNN, and ANN applying the same GLCM training and GLCM validation datasets. The results for classifying individual patches are shown in Table I.

TABLE I
PERFORMANCE OF CLASSIFYING INDIVIDUAL PATCHES.

Method	Accuracy	Precision	Recall	F1
SVM ($C = 10^3, \gamma = 1$)	80.8	79.1	86.0	82.4
KNN ($K = 15$)	73.2	74.2	75.0	74.6
ANN (iter = 10^3)	79.1	78.5	82.8	80.6

For each method in Table I, experiments are conducted under different hyper-parameter settings, and for each method, only the best performance is presented. The SVM equipped with RBF kernel exhibits the best performance in predicting image patches of rooftops and backgrounds. The corresponding cost and gamma are noted as 1000 and 1 respectively. Table II shows the performance of the audit model on the validation images. In this case, the audit model uses the multimodal classification presented in Fig. 4. The SVM-based audit classifier has outperformed the other variants by a significant margin in accuracy, recall, and F1-score. Therefore, we analyze SVM further as the most effective audit classifier and conducted 5 fold cross-validation. The resulting mean values of accuracy, precision, recall, and F1-score are computed as 86.6, 88.3, 95.7, and 91.8 respectively.

TABLE II
PERFORMANCE OF THE AUDIT MODEL.

Audit Classifier	Accuracy	Precision	Recall	F1
SVM ($C = 10^3, \gamma = 1$)	87.5	88.2	97.1	92.4
KNN ($K = 15$)	68.2	95.6	62.3	75.4
ANN (iter = 10^3)	79.5	91.8	81.2	86.2

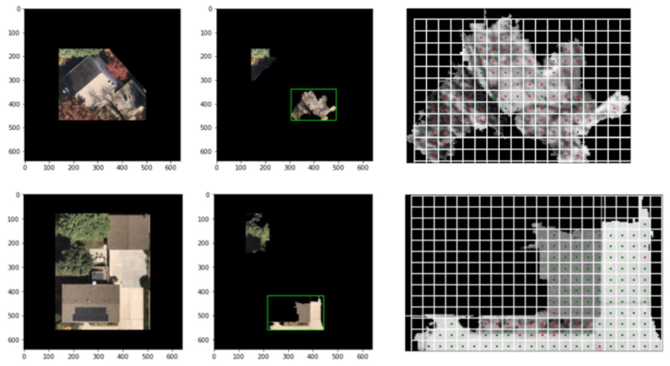


Fig. 5. Special cases of possible miss and unnecessary review. From left to right: image x, image ℓ, and a bag of patches. Top row: smooth texture and unsupportive color variance lead to a possible miss. Bottom row: rough texture and unsupportive color variance lead to an unnecessary review. (Original satellite images courtesy of Nearmap US, Inc.)

C. Discussion

Auditing of explainability is a type of sanity check for the original classifier. The major purpose of this auditing is to detect cases where the original classifier is likely to misclassify. However, this may lead to cases where validation by a human expert may be deemed unnecessary although validation could be beneficial or vice versa. In this section, we cover four cases with respect to the quality of the audit and

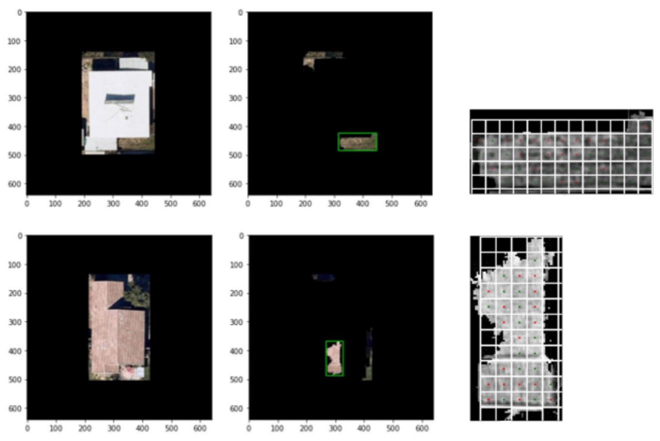


Fig. 6. Special cases of detection and no review. From left to right: image x, image ℓ, and a bag of patches. Top row: shadows or branches of trees obstruct the rooftop in the image ℓ. Bottom row: background very similar to some flatroofs confuse the classifiers. (Original satellite images courtesy of Nearmap US, Inc.)

the explanation for validating results by a human expert (Table III). In this discussion, rather than focusing on successful auditing, we provide one example per case. Fig. 5 and Fig. 6 show validation images, selected segments of the local explanation, and a respective bag of image patches. Any image patch predicted as *rooftop* is marked with a green pixel, and any image patch predicted as *background* is marked with a red pixel.

TABLE III
CASES OF VALIDATION.

Validation Case	Audit Result	Ground Truth
Recommended	Weak	Weak
Unnecessary	Satisfactory	Satisfactory
Missed	Satisfactory	Weak
Extraneous	Weak	Satisfactory

Case 1. Validation is recommended. The top row of Fig. 5 illustrates a local explanation indicating a view of a rooftop obstructed by branches or shadows of trees. In this case, the audit classifier technically classifies the segment of local explanation as *weak* since most image patches are similar to the ones that come from the background. Regardless of the original model classification, such an image requires additional review to avoid errors. Hence, validation is recommended.

Case 2. Validation is unnecessary. In the bottom row of Fig. 5, the audit classifier classifies this explanation as *satisfactory*. In this image, the image covers rooftop-like regions including the rooftop and the driveway. Since the driveways have similar textures as flat roofs or they are eligible regions to roof, this explanation is considered satisfactory. We should note that this auditing does not check the ability of the original classifier to distinguish a driveway from a flat roof. This is rather an indication that the classifier analyzes proper regions from the image. Since the classifier analyzes proper regions for determining the rooftop type, validation is unnecessary in this case.

Case 3. Validation is missed. The top row of Fig. 6 illustrates a segment of the local explanation revealing the background but classified incorrectly as *satisfactory*. Normally, it would be beneficial to validate this case by an expert regardless of the original model's classification.

Case 4. Validation is extraneous. The bottom row of Fig. 6 illustrates a segment of the local explanation revealing the rooftop but classified as *weak*. This leads to an unnecessary review by an expert.

V. Conclusion and Future Work

In this paper, we have analyzed auditing of explainability for automating the evaluation of machine learning systems. As the volume of data where these machine learning systems increase, a human expert can not check the explanation of each decision made by the system. Random or arbitrary checks are insufficient to guarantee an overall local explanation. We propose the *ExplainabilityAudit* system where the proposed system analyzes whether the right segments or components of images are processed by machine learning models to make the prediction. The concepts introduced in this paper reflect the first stage of ongoing research to develop *ExplainabilityAudit*. Our experimental results confirm that the suggested version of the *ExplainabilityAudit* is capable of predicting the reliability of the image recognition algorithm with a satisfactory recall of 95.7% indicating that 95.7% of the cases to be audited have been detected by the proposed system.

In the future, the possibility of replacing LIME with other explainability tools such as Grad-CAM is prominent. Also, the analysis should be further carried out for a multiclass image classification algorithm. Then, the *ExplainabilityAudit* would highlight the weaknesses of such a classifier with respect to different class labels. Moreover, rather than using a single explanation (largest superpixel), all the segments of the local explanations should be selected for making the bag of image patches. Adding more GLCM features and other metrics such as image histograms might help fix the issues with the audit classifier related to the sensitivity to roughness and color of texture. Instead of just producing audit labels, it would be possible to modify the audit model to generate a confidence value for the explanation.

Acknowledgment

The work was supported primarily by the Travelers Indemnity Company. The opinions, findings and conclusions or recommendations expressed in this material only reflect those of the authors in their individual capacities.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[2] ——, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[3] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–42, 2018.

[4] A. Rai, "Explainable ai: From black box to glass box," *Journal of the Academy of Marketing Science*, vol. 48, no. 1, pp. 137–141, 2020.

[5] C. Molnar, G. Casalicchio, and B. Bischl, "Interpretable machine learning–a brief history, state-of-the-art and challenges," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2020, pp. 417–431.

[6] M. Langer, K. Baum, K. Hartmann, S. Hessel, T. Speith, and J. Wahl, "Explainability auditing for intelligent systems: A rationale for multi-disciplinary perspectives," in *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*. IEEE, 2021, pp. 164–168.

[7] J. Wexler, M. Pushkarna, T. Bolukbasi, M. Wattenberg, F. Viégas, and J. Wilson, "The what-if tool: Interactive probing of machine learning models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 56–65, 2020.

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

[9] A. Amazon Web Services, "Aws sagemaker clarify," 2020. [Online]. Available: https://aws.amazon.com/sagemaker/clarify/

[10] M. T. Ribeiro, S. Singh, and C. Guestrin, """ why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[11] I. Tenney, J. Wexler, J. Bastings, T. Bolukbasi, A. Coenen, S. Gehrmann, E. Jiang, M. Pushkarna, C. Radebaugh, E. Reif, and A. Yuan, "The language interpretability tool: Extensible, interactive visualizations and analysis for NLP models," pp. 107–118, 2020. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-demos.15

[12] M. Korobov, "Explaining behavior of machine learning models with eli5 library," EuroPython, 2017, https://doi.org/10.5446/33771 $Lastaccessed: 27Sep$2021.

[13] Oracle Corporation, "Oracle skater," 2020. [Online]. Available: https://github.com/oracle/Skater

[14] M. R. Zafar and N. M. Khan, "Dlime: A deterministic local interpretable model-agnostic explanations approach for computer-aided diagnosis systems," *arXiv preprint arXiv:1906.10263*, 2019.

[15] S. M. Shankaranarayana and D. Runje, "Alime: Autoencoder based approach for local interpretability," in *International conference on intelligent data engineering and automated learning*. Springer, 2019, pp. 454–463.

[16] E. Lee, D. Braines, M. Stiffler, A. Hudler, and D. Harborne, "Developing the sensitivity of lime for better machine learning explanation," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, vol. 11006. International Society for Optics and Photonics, 2019, p. 1100610.

[17] S. Shi, X. Zhang, and W. Fan, "A modified perturbed sampling method for local interpretable model-agnostic explanation," *arXiv preprint arXiv:2002.07434*, 2020.

[18] X. Zhao, X. Huang, V. Robu, and D. Flynn, "Baylime: Bayesian local interpretable model-agnostic explanations," *arXiv preprint arXiv:2012.03058*, 2020.

[19] T. Löfstedt, P. Brynolfsson, T. Asklund, T. Nyholm, and A. Garpebring, "Gray-level invariant haralick texture features," *PloS one*, vol. 14, no. 2, p. e0212110, 2019.