Explaining Image Misclassification in Deep Learning via Adversarial Examples

Rami Haffar[0000−0002−0139−246X], Najeeb Moharram Jebreel[0000−0002−4911−3802], Josep Domingo-Ferrer[0000−0001−7213−4962], and David Sánchez[0000−0001−7275−7887]

Universitat Rovira i Virgili
Department of Computer Engineering and Mathematics
CYBERCAT-Center for Cybersecurity Research of Catalonia
UNESCO Chair in Data Privacy
Av. Països Catalans 26
43007 Tarragona, Catalonia
{rami.haffar,najeebmoharramsalim.jebreel,josep.domingo,david.sanchez}@urv.cat

Abstract. With the increasing use of convolutional neural networks (CNNs) for computer vision and other artificial intelligence tasks, the need arises to interpret their predictions. In this work, we tackle the problem of explaining CNN misclassification of images. We propose to construct adversarial examples that allow identifying the regions of the input images that had the largest impact on the CNN wrong predictions. More specifically, for each image that was incorrectly classified by the CNN, we implemented an inverted adversarial attack consisting on modifying the input image as little as possible so that it becomes correctly classified. The changes made to the image to fix classification errors *explain* the causes of misclassification and allow adjusting the model and the data set to obtain more accurate models. We present two methods, of which the first one employs the gradients from the CNN itself to create the adversarial examples and is meant for model developers. However, end users only have access to the CNN model as a black box. Our second method is intended for end users and employs a surrogate model to estimate the gradients of the original CNN model, which are then used to create the adversarial examples. In our experiments, the first method achieved 99.67% success rate at finding the misclassification explanations and needed on average 1.96 queries per misclassified image to build the corresponding adversarial example. The second method achieved 73.08% success rate at finding the explanations with 8.73 queries per image on average.

Keywords: Explainability · Deep learning · Image classification · Adversarial examples · Convolutional neural networks

1 Introduction

The use of deep learning, and of convolutional neural networks (CNNs) in particular, has brought great advances in computer vision [11] and many other artificial

intelligence (AI) endeavors. Although CNNs can achieve high accuracy in classification, detection, and segmentation tasks, they are black-box models. This means that their predictions do not come with explanations or justifications and, therefore, it is not possible for humans to understand how decisions were made. To avoid blind algorithm-based decisions, AI models should be explainable [2]. Explainability is not only an ethical principle but also a legal requirement set out in the European General Data Protection Regulation (GDPR) [6]. The lack of explanations about the decisions made by CNNs is a problem both for developers who train such networks and for the citizens affected by their decisions:

- Developers want to know how a decision is made, to ensure that the AI model takes into account the correct features of the input during the training phase. In some cases, it may happen that wrong features are used to make decisions, such as in the well-known example of [14] where a dog-like animal is classified as a wolf if the image has a snow background and as a husky dog if the image has a grass background because in the training pictures all the wolves were displayed in a snowy landscape and the huskies were not.
- Citizens are affected by a growing number of automated decisions: credit granting, insurance premiums, medical diagnoses, etc. For that reason, legal regulations [6] and ethics guidelines [4, 16] have appeared that assert the citizen's right to an explanation on every automated decision affecting her. Lacking such explanations, even *de iure* democracies risk becoming *de facto* AI-driven authoritarian societies.

In computer vision, the interpretations of CNN predictions are usually presented as saliency maps. Those maps suggest specific regions in the images that are most important in the decision made by the CNN [7].

Contributions and plan of this paper

For the generation of explanations to be scalable and efficient, it must be automated. In this work, we present two methods that explain CNN-based image classification by identifying the features that were most influential in the CNN predictions. The first method assumes access to the gradients of the CNN and is meant for model developers. The second method treats the model as a black box and, therefore, assumes that the party generating the explanations, such as a model end user, only has access to the model predictions.

Both methods leverage adversarial examples [13] to generate explanations. While the first methods computes adversarial examples by directly employing the CNN gradients, the second approach builds a simpler surrogate model to estimate the gradients of the original model, and then uses these estimated gradients to obtain adversarial examples. More specifically, for each image that was incorrectly classified by the CNN, we implemented an inverted adversarial attack consisting in modifying the input image as little as possible so that it becomes correctly classified. The changes made to the image to fix classification errors highlight the regions that had the highest influence in the decisions and

thus *explain* the causes of model misclassification. By identifying the causes of wrong predictions, one may tailor the model or the training data to improve the classification accuracy.

The remainder of this paper is organized as follows. Section 2 discusses related works devoted to explaining CNNs. Section 3 describes our methods for explaining CNN-based image misclassification from adversarial examples. Experimental results are reported in Section 4. Finally, in Section 5 we gather conclusions and sketch future research lines.

2 Related work

Several methods have been proposed to interpret CNN predictions in image classification. They attempt to link inputs to outputs to identify the regions in the image that have the highest impact on the classification decision.

One of the most commonly used approaches to generate explanations is to calculate the influence of the image features by back-propagating the decision score across all layers of the network. Works that follow this approach are XRAI [10], Guided Backprop [20], Gradient Input [17], SmoothGrad [19] and GradCAM [15]. This approach is fast and the methods following it usually require a fixed number of queries to the black-box model. However, they need access to all the internal layers of the model. Also, the computational cost of those methods is high because a second back-propagation is necessary.

It is also possible to locally train a simpler model, a.k.a. surrogate model, to approximate the black-box behavior and obtain an explanation of the black-box model [14]. This approach requires that the surrogate model be simple and understandable to humans. Unfortunately, the surrogate model is often much inferior to the black-box model in terms of accuracy, and hence the explanations provided by the former are not very reliable [8].

Another approach, known as perturbation-based, is to modify the input images and measure the effect of this change on the black-box prediction [5, 3, 22]. Perturbation-based methods allow directly detecting the regions of the images that had the highest impact on the predictions. Typically, these methods require multiple queries to the black box in order to interpret its predictions, which makes them slow [23].

The methods we propose in this paper fall in the perturbation-based category. By using gradient-based adversarial examples to add the perturbation to the original image, we are able to minimize the required number of queries to the model and we keep the computational cost at a minimum. Also, thanks to a surrogate model with reasonable complexity and accuracy compared to the black-box model, we can generate explanations just from the black-box prediction, without knowing any details on the black box's internal layers.

3 Our proposals

We focus on generating explanations for the wrong predictions made by CNNs by identifying the regions of the input images that had the highest influence on those predictions. To this end, we need a way to modify the input images towards the correct classification while keeping the number of required queries and the computational cost at a minimum. Our choice is to use gradient-based adversarial examples [21]. Specifically, we add minimal perturbations to incorrectly classified images to create correctly classified adversarial examples. Then, by comparing the original image with the modified image, we can find the regions that had the highest impact on the wrong black-box predictions.

3.1 Adversarial examples

An adversarial example is a sample from the same distribution as the original data in which small, intentional perturbations of its features cause an AI model to change its prediction [12]. Adversarial examples can be used to alter predictions of a variety of machine learning models, including state-of-the-art neural networks [21]. Even though adversarial examples are usually employed to cause the AI models to produce wrong predictions, in this work we use them the other way around: to correct wrongly classified samples.

To create adversarial examples we used the gradient-based optimization approach proposed in [21], in which we set the target to be the correct label for the wrongly predicted samples. The adversarial examples are created by minimizing the following function with respect to r:

$$loss(f(x+r), l) + \epsilon \cdot |r|, \tag{1}$$

where f is the AI classifier, x is the original image, r is the perturbation added to the pixels of x to create the perturbed image that constitutes the adversarial example, l is the target class label and ϵ is used to balance the distance between images and the distance between predictions. The smaller ϵ, the more similar is the created perturbed image to the original image. To minimize the loss function in Equation (1), the party that computes it needs access to the model gradients.

3.2 Explaining model predictions on the developer's side

To explain the wrong predictions made by the model on the developer's side, we consider that the developer has full access to the CNN and, more specifically, to the gradients of the model.

As shown in Algorithm 1, first the developer splits the input images into training and testing sets and trains the model with the training images. Then, in the testing phase the developer keeps track of all the wrongly classified images. For each of these images, she tries to find the closest adversarial example that is correctly classified. The developer does this in the following way: i) she calculates the value of the loss function between the model prediction and the

correct prediction; ii) she calculates the gradients of the model according to the image and the loss value; 3) she modifies the image according to the gradients and the perturbation ratio ϵ. These steps are repeated until the adversarial example is obtained or ϵ exceeds the α value signaling the termination condition (in the latter case the image misclassification cannot be explained). The final step consists in comparing each original image with its corresponding adversarial example. To draw a saliency map that identifies the features that caused wrong predictions, Algorithm 2 prescribes that pixels in perturbations with values smaller than $q3 + iqr \cdot \tau$, where $q3$ is the third quartile of perturbations, iqr is their interquartile range and $\tau > 0$ is relaxation parameter, are neglected because they do not identify regions of interest, whereas the remaining pixels are multiplied by $\beta > 1$ to boost them in the saliency map.

Algorithm 1 Explaining the model predictions on the developer's side

input: Data set X, CNN model *model*
$Train\_X, Test\_X \leftarrow Split\_Train\_Test(X)$
$model \leftarrow Train\_Model(Train\_X)$
$perturbations \leftarrow \{\}$
for *i in* $Test\_X$ **do**
  $model\_prediction \leftarrow model.predict(Test\_X[i])$
  $\epsilon \leftarrow 0.1$
  while $model\_prediction \neq correct\_prediction$ OR $\epsilon < \alpha$ **do**
    $loss \leftarrow loss\_function(model\_prediction, correct\_prediction)$
    $gradients \leftarrow get\_model\_gradients(Test\_X[i], loss)$
    $perturbed\_image \leftarrow Test\_X[i] - \epsilon \cdot gradients$
    $model\_prediction \leftarrow model.predict(perturbed\_image)$
    $\epsilon \leftarrow \epsilon + 0.1$
  end while
  if $model.prediction = correct\_prediction$ **then**
    $perturbations[i] \leftarrow perturbed\_image - Test\_X[i]$
  else
    $perturbations[i] \leftarrow NIL$
  end if
end for
return *perturbations*

3.3 Explaining model predictions on the user's side

End users should have the right to obtain explanations about predictions made by the AI models that concern them. However, for end users, the model is a black box and they only have access to the model predictions. Therefore, they must create their own local explanations. In our work, we considered that the user who wants to generate explanations of an AI model must have enough data to train a simpler CNN model, a.k.a. a surrogate model. It is shown in [9] that

Algorithm 2 Drawing the saliency maps

```
input: perturbations
q1, q3 = get_quartiles_of_non_NIL_perturbations(perturbations)
iqr ← q3 − q1
for i such that perturbations[i] ≠ NIL do
  for pixel in perturbations[i] do
    if perturbations[i][pixel] < q3 + iqr · τ then
      perturbations[i][pixel] ← 0
    else
      perturbations[i][pixel] ← perturbations[i][pixel] · β
    end if
  end for
  Draw(perturbations[i])
end for
```

knowledge of one or more models can be compressed into another, less complex model, which allows us to estimate the gradations of the original model using a surrogate model.

The method we propose is formalized in Algorithm 3. First, the user splits the data she has into training and testing data sets. Then she builds a surrogate model by using the local training data set. Afterwards, she uses the test data set to identify the wrong predictions to be explained. Finally, she generates the adversarial examples in a similar way as in Algorithm 1. The only difference is that the gradients from the surrogate model will be used instead of the gradients from the original model.

4 Experimental results

We tested the two proposed methods introduced above on the gender classification data set[1] from the Kaggle website. This data set consists of cropped RGB images of male and female faces. The training data set contains 23,200 female images and 23,800 male images. The validation data set contains 5,800 images in each class. The images are rectangular, but not all of them are of the same size. Thus, we first resized all the images to 100 x 100 pixels.

4.1 Explanations for the developer

To test Algorithm 1 we used the CNN shown in Figure 1 with four Conv blocks followed by six fully connected layers. Each Conv block contained two convolutional layers and a max-pooling layer. The output depths for Conv blocks were, respectively, 64, 128, 256 and 512. The numbers of nodes of fully connected layers were, respectively, 2048, 1024, 512, 128, 32 and 2. We trained the model for 20 epochs, with a batch size 64 and a learning rate 0.001. The test accuracy was 96.3%.

[1] https://www.kaggle.com/cashutosh/gender-classification-dataset

Algorithm 3 Explaining the model predictions on the user's side

input: local data X_local , black-box model $black\_box$, surrogate model $local\_surrogate$
$Train\_X\_local, Test\_X\_local \leftarrow Split\_Train\_Test(X\_local)$
$local\_surrogate \leftarrow Train\_Black\_Box(Train\_X\_local)$
for i *in* $Test\_X\_local$ **do**
  $black\_box\_prediction \leftarrow black\_box.predict(Test\_X\_local[i])$
  $\epsilon \leftarrow 0.1$
  while $black\_box\_prediction \neq correct\_prediction$ OR $\epsilon < \alpha$ **do**
    $local\_prediction \leftarrow local\_surrogate.predict(Test\_X\_local[i])$
    $loss \leftarrow loss\_function(local\_prediction, correct\_prediction)$
    $gradients \leftarrow get\_surrogate\_gradients(Test\_X\_local[i], loss)$
    $perturbed\_image \leftarrow Test\_X\_local[i] - \epsilon \cdot gradients$
    $black\_box\_prediction \leftarrow black\_box.predict(perturbed\_image)$
    $\epsilon \leftarrow \epsilon + 0.1$
  end while
  if $black\_box\_prediction = correct\_prediction$ **then**
    $perturbations[i] \leftarrow perturbed\_image - Test\_X\_local[i]$
  else
    $perturbations[i] \leftarrow NIL$
  end if
end for
return $perturbations$

The number of misclassified images in the test data set was 301, for which our method created adversarial examples. 300 of the 301 adversarial examples were classified into the correct labels, which corresponds to a success rate 99.67%, with an average number 1.96 queries per image. Figure 2 shows four examples of the explanations created by Algorithm 1, in the form of saliency maps highlighting the differences between original images and adversarial examples.

In Figure 2, we can see that perturbations added to the original images to create the adversarial examples are not noticeable to the naked eye. Nevertheless, the explanations resulting from Algorithm 1 in the form of saliency maps shed light on the most important regions that caused the wrong predictions. For example, in Image 1 the important pixels were those around the eyes and the edge of the nose. In Image 2 the causes of wrong classification were also found in the eyes and the nose in addition to the left cheek. In Image 3, the causes of misclassification were mainly the left eye and the edge of the right eye in addition to parts of the covered forehead. The most relevant regions for Image 4 were the left eye, the edge of the nose and the left cheek.

4.2 Explanations for the user

Testing the performance of Algorithm 3 tells whether the user is capable of creating model explanations locally. We assumed the user's local data consisted of a random 10% sample of the training data described in the previous section. With

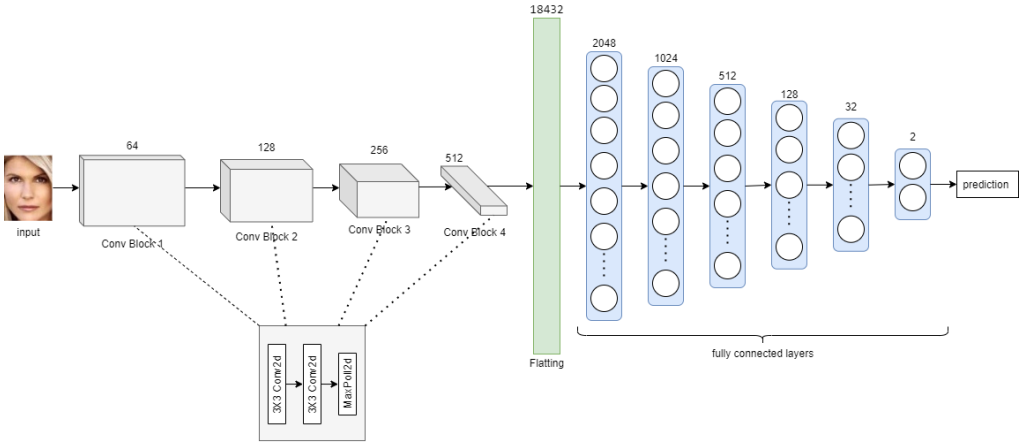

Fig. 1. Architecture of the CNN used as original model in the experiments

these local data, the user trained her surrogate model. The black-box model was the same CNN described in the previous section, whereas the local surrogate model built by the user consisted of a CNN with three Conv blocks followed by four fully connected layers. Each Conv block contained two convolutional layers and a max-pooling layer. The output depths for Conv blocks were, respectively, 64, 256 and 512. The numbers of nodes of fully connected layers were, respectively, 1024, 256, 32 and 2. We trained the model for 50 epochs, with a batch size 64 and a learning rate 0.001. The test accuracy for the surrogate model was 87.78%.

The complete test data set was used to generate explanations. Therefore, we got the same 301 misclassified images. Following Algorithm 3, explanations were obtained as follows: i) obtain the prediction and gradients of the local surrogate model; ii) create the adversarial example using the gradients of the surrogate model; iii) test whether the adversarial example was correctly predicted by using the original black-box model; iv) draw the saliency map. Out of the 301 adversarial examples, the original black-box model correctly classified 220 images, which corresponds to a 73.08% success rate. The average number of queries to the original CNN model per image required to create the adversarial example was 8.73.

Figure 3 shows the same four samples of Figure 2 but with saliency maps that were locally generated using the gradients of the surrogate model. As in the previous test, the differences between the adversarial examples and the original images are not noticeable to the naked eye. However, the most relevant regions of the images are similar to those obtained with Algorithm 1: in the four images the same regions highlighted by Algorithm 1 are also highlighted here, even though in a less focused way due to the less accurate surrogate model.

The number of queries per image needed to create adversarial examples with Algorithm 1 was lower than with Algorithm 3: 1.96 vs 8.73. The reason is that

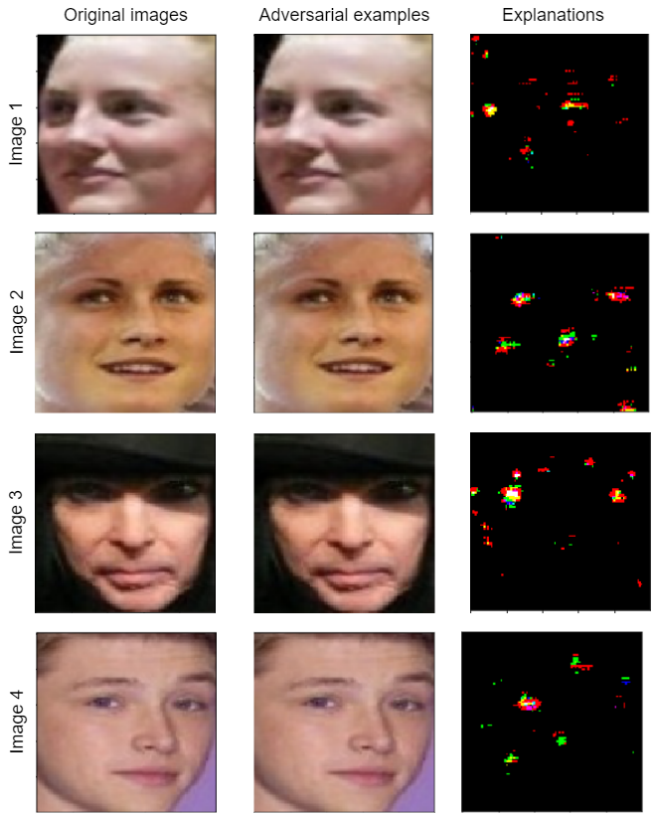

Fig. 2. Four examples of explanations generated on the developer's side

the former algorithm uses the gradients from the original model, whereas the latter uses the gradients from the surrogate model. Hence, the generation of adversarial examples is less accurate with the second algorithm. However, both algorithms were successful in generating explanations for the wrong predictions of the original model and both highlighted the same regions as important.

The explanations provided by our methods can help model developers to identify the weaknesses of the data sets used to train the model. Specifically, for the gender classification data set, the eye regions are highlighted in most saliency maps as important regions. This suggests that classification accuracy may be improved by training on images where the eye region is clear. For model users, it is important to know which features influenced the predictions since some of the black boxes may be artificially biased or employ features that may discriminate some minorities [1]. Beyond face classification, an even more crucial application could be to help understand medical diagnoses [18].

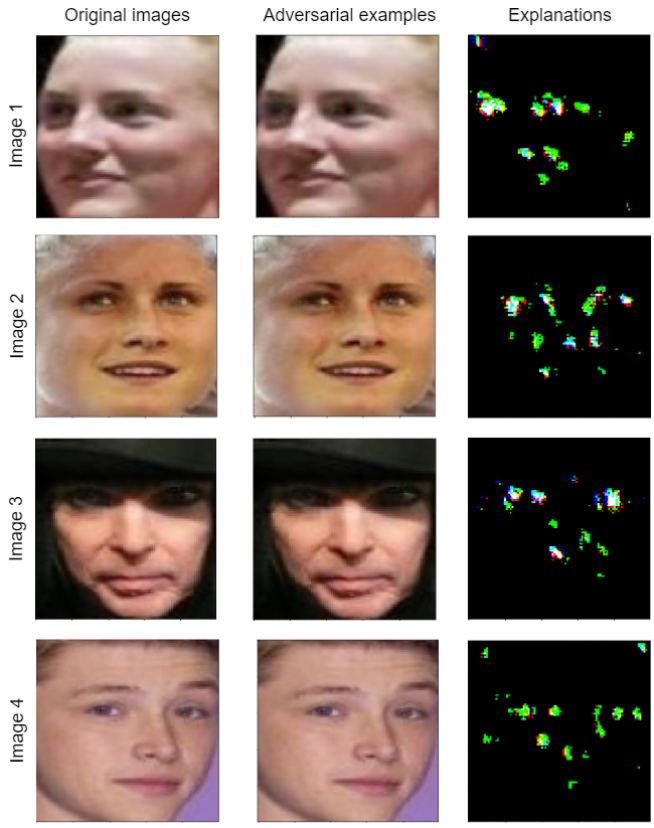

Fig. 3. Four examples of explanations generated on the user's side

5 Conclusions and future research

We have presented two methods employing gradient-based adversarial examples to obtain explanations of the predictions of CNNs in image classification.

We have reduced the number of queries needed to create the adversarial examples by adding targeted perturbations to change the predictions for each image. In our experimental work, developer-side Algorithm 1 required only 1.96 queries per image, whereas user-side Algorithm 3 needed 8.73 queries per image.

The two proposed algorithms showed promising results to explain misclassification by CNNs. Both produced similar explanations on the same samples. Algorithm 1 had a higher success rate (99.67%) thanks to using the gradients of the original model, whereas Algorithm 3 had a lower success rate (73.08%) due to using a surrogate.

As future work, we plan to test the performance of our approach on data that are not identically and independently distributed. We also plan to tailor the

generation of adversarial examples to highlight regions of interest in correctly classified images.

Acknowledgments

We acknowledge support from the European Commission (projects H2020-871042 "SoBigData++" and H2020-101006879 "MobiDataLab"), the Government of Catalonia (ICREA Acadmia Prizes to J. Domingo-Ferrer and D. Snchez, and grant 2017 SGR 705) and from the Spanish Government (projects RTI2018-095094-B-C21 "Consent" and TIN2016-80250-R "Sec-MCloud"). The authors are with the UNESCO Chair in Data Privacy, but the views in this paper are their own and are not necessarily shared by UNESCO.

References

1. Azad, R., Fayjie, A.R., Kauffmann, C., Ben Ayed, I., Pedersoli, M., Dolz, J.: On the texture bias for few-shot cnn segmentation. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 2674–2683 (2021)
2. Blanco-Justicia, A., Domingo-Ferrer, J., Martínez, S., Sánchez, D.: Machine learning explainability via microaggregation and shallow decision trees. Knowledge-Based Systems **194**, 105532 (2020)
3. Carter, B., Mueller, J., Jain, S., Gifford, D.: What made you do this? understanding black-box decisions with sufficient input subsets. In: The 22nd International Conference on Artificial Intelligence and Statistics. pp. 567–576. PMLR (2019)
4. European Commission's High-Level Expert Group on Artificial Intelligence: Draft Ethics Guidelines for Trustworthy AI (2019), https://ec.europa.eu/futurium/en/ai-alliance-consultation
5. Fong, R.C., Vedaldi, A.: Interpretable explanations of black boxes by meaningful perturbation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3429–3437 (2017)
6. GDPR: General Data Protection Regulation, Regulation EU 2016/679 of the European Parliament and of the Council of 27 April 2016. Official Journal of the European Union. Available at: http://ec. europa. eu/justice/data-protection/reform/files/regulation_oj_en. pdf (accessed 20 September 2017) (2016)
7. Ghorbani, A., Abid, A., Zou, J.: Interpretation of neural networks is fragile. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 3681–3688 (2019)
8. Haffar, R., Domingo-Ferrer, J., Sánchez, D.: Explaining misclassification and attacks in deep learning via random forests. In: International Conference on Modeling Decisions for Artificial Intelligence. pp. 273–285. Springer (2020)
9. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
10. Kapishnikov, A., Bolukbasi, T., Viégas, F., Terry, M.: Xrai: Better attributions through regions. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4948–4957 (2019)
11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25**, 1097–1105 (2012)

12. Molnar, C.: Interpretable machine learning. Lulu. com (2020)
13. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 427–436 (2015)
14. Ribeiro, M.T., Singh, S., Guestrin, C.: " 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 (2016)
15. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE international conference on computer vision. pp. 618–626 (2017)
16. Shahriari, K., Shahriari, M.: IEEE standard review. Ethically aligned design: a vision for prioritizing human wellbeing with artificial intelligence and autonomous systems. In: 2017 IEEE Canada International Humanitarian Technology Conference (IHTC). pp. 197–201. IEEE (2017)
17. Shrikumar, A., Greenside, P., Shcherbina, A., Kundaje, A.: Not just a black box: Learning important features through propagating activation differences. arXiv preprint arXiv:1605.01713 (2016)
18. Singh, V.K., Abdel-Nasser, M., Rashwan, H.A., Akram, F., Haffar, R., Pandey, N., Sarker, M.M.K., Kohan, S., Guma, J., Romani, S., et al.: Mass detection in mammograms using a robust deep learning model. In: CCIA. pp. 365–372 (2019)
19. Smilkov, D., Thorat, N., Kim, B., Viégas, F., Wattenberg, M.: Smoothgrad: removing noise by adding noise. arXiv preprint arXiv:1706.03825 (2017)
20. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. In: International Conference on Machine Learning. pp. 3319–3328. PMLR (2017)
21. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
22. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: European conference on computer vision. pp. 818–833. Springer (2014)
23. Zintgraf, L.M., Cohen, T.S., Adel, T., Welling, M.: Visualizing deep neural network decisions: Prediction difference analysis. arXiv preprint arXiv:1702.04595 (2017)