The Need for Empirical Evaluation of Explanation Quality

Nicholas Halliwell, Fabien Gandon, Freddy Lecue, Serena Villata

The Need for Empirical Evaluation of Explanation Quality

Nicholas Halliwell,[1] **Fabien Gandon,** [1] **Freddy Lecue,** [1, 2] **Serena Villata** [1]

[1] Inria, Université Côte d'Azur, CNRS, I3S, France
[2] CortAIx, Thales, Montreal, Canada
nicholas.halliwell@inria.fr, fabien.gandon@inria.fr, freddy.lecue@inria.fr, serena.villata@inria.fr

Abstract

Prototype networks (Li et al. 2018) provide explanations to users using a prototype vector; that is, a vector learned by the network representing a "typical" observation. In this work, we propose an approach that identifies relevant features in the input space used by the Prototype network. We find however that empirical evaluation of explanation quality is difficult without ground truth explanations. We include a discussion about developing methods for generating explanations, identifying when one explanation method is preferable to another, and the complications that arise when measuring explanation quality.

1 Introduction

Deep learning models are used to serve automated decisions in settings such as banks, insurance, and health care. These models are typically treated as a black box, where no insight is given as to how they make decisions. This lack of transparency has hindered adoption of these models into production. Much research has been devoted to developing algorithms, or explanation methods, to interpret their predictions.

Indeed there are many approaches for generating post-hoc explanations. Feature importance methods (Lundberg and Lee 2017; Ribeiro, Singh, and Guestrin 2016; Kim et al. 2018), where relevant dimensions are identified and assigned a score to rank its importance relative to the other dimensions. For image data, saliency maps (Simonyan, Vedaldi, and Zisserman 2014; Springenberg et al. 2015; Bach et al. 2015; Selvaraju et al. 2016; Shrikumar, Greenside, and Kundaje 2017; Shrikumar et al. 2016; Zeiler and Fergus 2014; Smilkov et al. 2017; Sundararajan, Taly, and Yan 2017; Montavon et al. 2017) identify relevant pixels in the input image. Counterfactual explanations (Wachter, Mittelstadt, and Russell 2017) give the smallest possible perturbation to the given input that will change the prediction to a desired target outcome. Lastly, prototype explanations (Chen et al. 2019; Li et al. 2018; Ming et al. 2019) learn a continuous vector that represents a "typical" training example, where explanations are given based on their relative distance to a prototype vector.

The Prototype network architecture from Li et al. (2018) combines an autoencoder with a prototype layer, where each observation in the training set is classified based on its distance to a prototype vector. The encoded input from the autoencoder is used as features for predictions downstream. The prototype vectors learned by this network are defined as typical observations in the training set, and, because they are learned in the same space as the encoded input, they can be mapped back into the original input space for visualization using the decoder. Explanations are given in the form of a most similar prototype vector. The specific architecture of this network allows us to further develop and improve the types of explanations generated post hoc.

In this paper, we expand the type of explanations generated by the Prototype network to identify relevant features in the input space. Due to the architecture of this network, the latent features learned by the model can be exploited to identify relevant input space features. We make use of the network's encoded input to randomly set latent features to zero, and use the network's decoder to determine which input space values changed the most. Finally, this work allows us to open a general discussion about generating explanations, identifying when one explanation method is preferable to another, and the complications that arise when measuring explanation quality.

2 Prototype Network

This section provides necessary background information on the Prototype network from Li et al. (2018), including the architecture and loss function.

Architecture Details

The Prototype network architecture can be visualized in Figure 1. It consists of an autoencoder (the encoder defined as $f : \mathbb{R}^p \to \mathbb{R}^q$ and the decoder, defined as $g : \mathbb{R}^q \to \mathbb{R}^p$), a prototype layer $p : \mathbb{R}^q \to \mathbb{R}^m$, and a dense (fully-connected) layer $w : \mathbb{R}^m \to \mathbb{R}^K$ that feeds into a softmax layer. The prototype layer takes as input encoded training points, denoted $f(\mathbf{x}_i)$, and computes the L^2 distance between $f(\mathbf{x}_i)$ and m prototype vectors, denoted $\mathbf{p}_1, \ldots, \mathbf{p}_m \in \mathbb{R}^q$. The overall network is given by $h : \mathbb{R}^q \to \mathbb{R}^K$. In this prototype network architecture, observations are classified based on their distance to a prototypical observation, and the loss function ensures that each prototype vector is similar to an encoded training point. We denote the data set $D =$

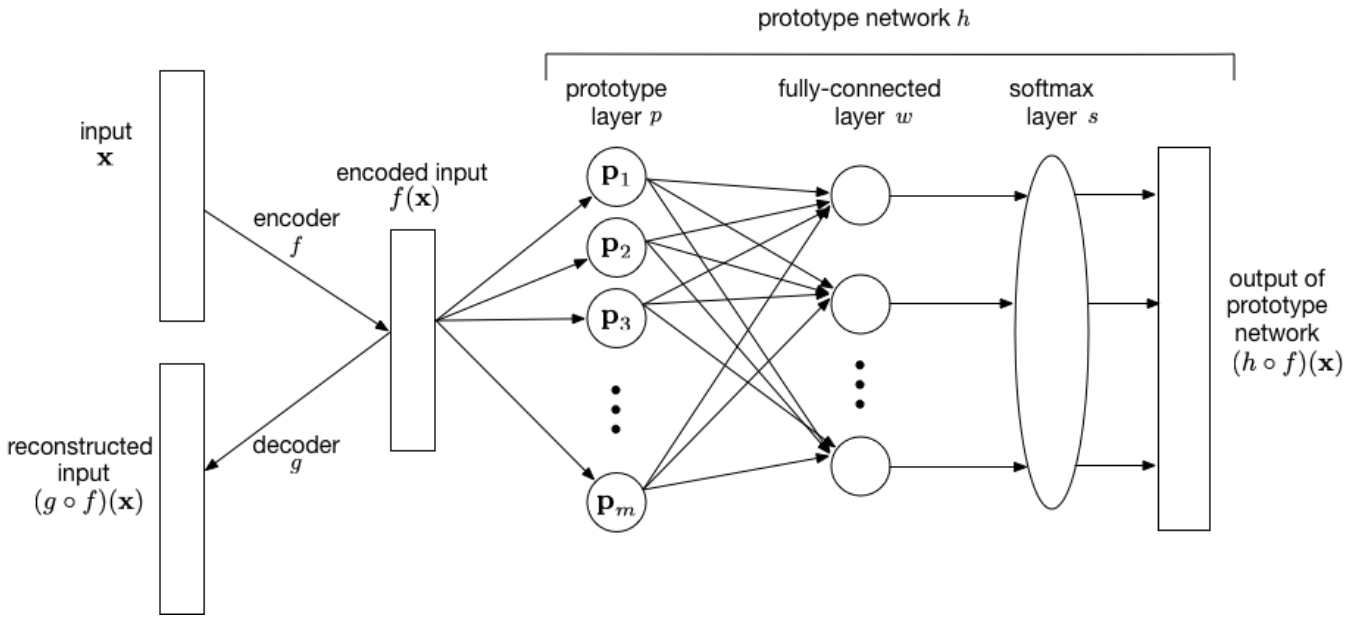


Figure 1: Prototype Network Architecture (Li et al. 2018).

$\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $y_i \in \{1, \dots K\}$, and K being the number of classes.

Loss Function

The loss function given by Li et al. (2018) is broken down into the following four parts below:

$$E(h \circ f,\ D) = \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} -\mathbb{1}[y_i = k] log((h \circ f)_k(\mathbf{x}_i)) \tag{1}$$

$$R(g \circ f,\ D) = \frac{1}{n} \sum_{i=1}^{n} ||(g \circ f)(\mathbf{x}_i) - \mathbf{x}_i||_2 \tag{2}$$

Two regularization terms are used, i.e., R_1, which forces each prototype vector to be as close as possible to one encoded training point, and R_2, which forces every encoded training point to be as close as possible to one prototype vector.

$$R_1(\mathbf{p}_1, \dots, \mathbf{p}_m,\ D) = \frac{1}{m} \sum_{j=1}^{m} \min_{i \in [1,n]} ||\mathbf{p}_j - f(\mathbf{x}_i)||_2 \tag{3}$$

$$R_2(\mathbf{p}_1, \dots, \mathbf{p}_m,\ D) = \frac{1}{n} \sum_{i=1}^{n} \min_{j \in [1,m]} ||f(\mathbf{x}_i) - \mathbf{p}_j||_2 \tag{4}$$

The complete loss function is given by

$$\begin{aligned} L((f, g, h), D) = E(h \circ f,\ D) + \lambda_0 R(g \circ f,\ D) + \\ \lambda_1 R_1(\mathbf{p}_1, \dots, \mathbf{p}_m,\ D) + \lambda_2 R_2(\mathbf{p}_1, \dots, \mathbf{p}_m,\ D) \end{aligned} \tag{5}$$

where λ_0, λ_1, λ_2 are hyperparameters.

3 Proposed Approach

The encoder function f maps a p dimensional vector to a q dimensional vector where $p > q$. This encoded input contains relevant information for classification, as it is used as features downstream, and is using a lower dimensional representation of the input data. Identifying relevant information in the encoded latent space should provide further insight into how the model is making decisions. For some observation $\mathbf{x}$ we want an explanation for, we encode the input using the Prototype network's encoder f. We then make m copies of the encoded input $f(\mathbf{x})$, and apply m different masks element-wise. Each mask, denoted $\mathbf{m}_i$, is the same dimensions as the encoded input $f(\mathbf{x})$, where each element of a mask is assigned a 1 with 90% probability and a 0 with 10% probability. The element-wise product is then averaged across the m masks, given by:

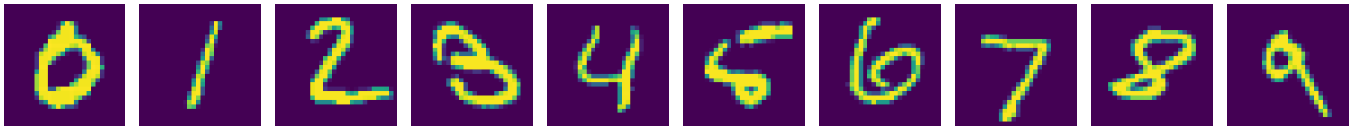
$$\hat{f}(\mathbf{x}) = \frac{1}{m} \sum_{i=1}^{m} f(\mathbf{x}_i) \odot \mathbf{m}_i. \tag{6}$$

The result $\hat{f}(\mathbf{x})$ is then decoded by the Prototype network's decoder g for visualization, given by:

$$\hat{g} = g(\hat{f}(\mathbf{x})). \tag{7}$$

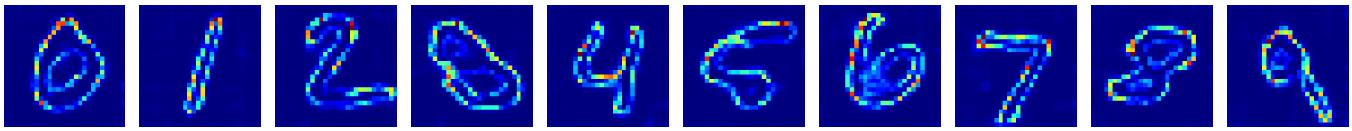
To identify the relevant dimensions in the input space, the input is mapped through the encoder and then decoded, denoted $g(f(\mathbf{x}))$. We then compute the absolute difference between the decoded input and the decoded masked input given by:

$$\mathbf{x}^* = |\hat{g} - g(f(\mathbf{x}))|. \tag{8}$$



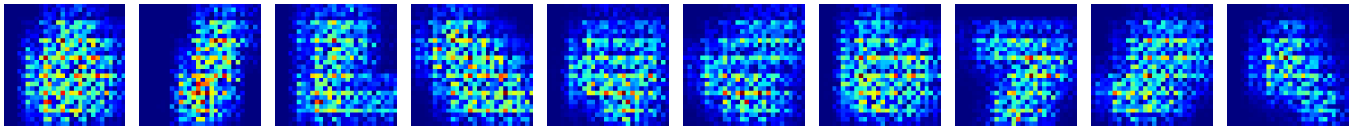
(a) 0 (b) 1 (c) 2 (d) 3 (e) 4 (f) 5 (g) 6 (h) 7 (i) 8 (j) 9

Figure 2: MNIST Images



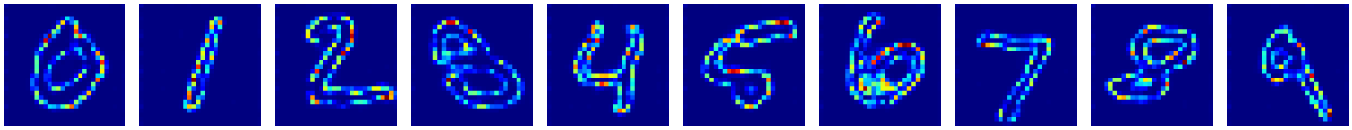
(a) 0 (b) 1 (c) 2 (d) 3 (e) 4 (f) 5 (g) 6 (h) 7 (i) 8 (j) 9

Figure 3: Saliency maps: Proposed approach



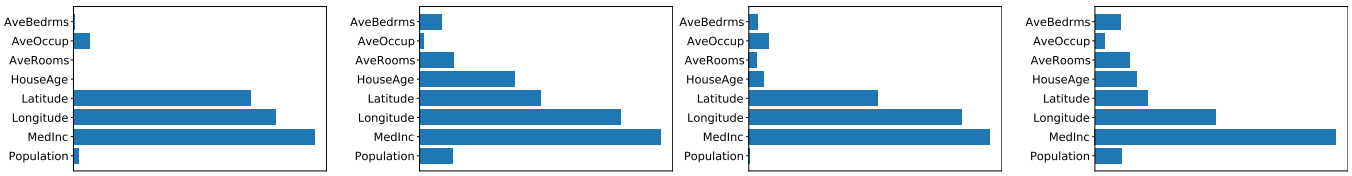
(a) 0 (b) 1 (c) 2 (d) 3 (e) 4 (f) 5 (g) 6 (h) 7 (i) 8 (j) 9

Figure 4: Saliency maps: Proposed approach-randomly initialized untrained network



(a) 0 (b) 1 (c) 2 (d) 3 (e) 4 (f) 5 (g) 6 (h) 7 (i) 8 (j) 9

Figure 5: Saliency maps: Proposed approach-network trained on randomly permuted labels



(a) Lime features: Observation 1 (b) Proposed features: Observation 1 (c) Lime features: Observation 2 (d) Proposed features: Observation 2

Figure 6: Explanations generated by Lime and the proposed approach on California Housing dataset.

where $\mathbf{x}^*$ gives the feature importance scores of $\mathbf{x}$ for each dimension. Here the absolute difference gives the features in the input space with the largest change. Code for this work is available online.[1]

4 Experiments

Image Data

With image data, we have the ability to visualize the explanation. We train a Prototype network on the MNIST dataset (LeCun et al. 1998) with 3 encoding layers, 3 decoding layers, 1 prototype layer, and 1 fully connected layer. This model learns 10 prototype vectors (one for each class), achieving 99.1% accuracy on the test set.

Figure 3 shows saliency maps of the proposed approach for each image in Figure 2. We can see that the proposed approach produces saliency maps that outline the digit in the original image. We perform the model parameter randomization and data randomization test (Adebayo et al. 2018). The model parameter randomization test generates saliency maps from a model with untrained, random parameters. The resulting saliency maps should be random noise. The data

[1] https://github.com/halliwelln/prototype-explanations/

randomization test trains a model where the training labels have been randomly shuffled. Like the model parameter randomization test, the resulting saliency maps should be random noise and the end user should not be able to determine the object in the image. Figure 4 shows saliency maps from an untrained Prototype network with randomly initialized parameters (model parameter randomization test). Figure 5 shows saliency maps for a model trained on random labels (data randomization test). From these figures we can see the proposed approach passes the model parameter randomization test but fails the data randomization test. In other words, the proposed approach to generating explanations is not providing insight into what the model has learned.

Tabular Data

We demonstrate our approach on a well known tabular dataset, the California Housing dataset (Pace and Barry 1997). Here, we are tasked with determining if houses should be sold above or below the median price. We train a Prototype network on the California Housing dataset with 2 encoding layers, and 2 decoding layers, 1 prototype layer, and 1 fully connected layer. This model learns 2 prototype vectors, achieving 84.2% accuracy on the test set.

Figure 6 compares relevant features identified by Lime (Ribeiro, Singh, and Guestrin 2016) to our proposed approach for selected observations. For both observations, we can see that the top 3 dimensions with the highest attribution scores are the same for both explanation methods. Although both explanations are similar, they are not exactly equal. From these examples, which explanation method is actually displaying what the model has learned? In other words, which explanation method is preferable to the other? These questions are difficult to answer without ground truth explanations to quantitatively compare against.

5 Discussion

From the experiments on tabular and image data, we found our approach produced what looked like faithful explanations on both types of data. After using the robustness tests from Adebayo et al. (2018) on an image dataset, we were able to determine that this was not the case. For image data, we have the ability to visually verify any explanation generated in the input space. With tabular data, we do not have this luxury. Depending on the type of data used for experimentation, researchers can be mislead into thinking the explanations their model is generating are faithful because they are similar to a state-of-the-art method. With ground truth explanations, researchers would not have to rely on previous state-of-the-art explanation methods to determine if their approach is generating faithful explanations.

In general, this is a common problem in the field of XAI. When a new explanation method is proposed, researchers often show several "good looking" examples to display to the reader the capability of the proposed method. Comparisons against a state-of-the-art method typically involve a small number of cherry-picked examples to demonstrate the ability of an explanation method. This can be misleading. Indeed a small number of selected examples does not truly represent how the explanation method is performing on the entire test set. As we demonstrated on the tabular dataset, our proposed approach can compete with Lime on "selected" examples, however, this is not conclusive evidence that this explanation method is preferable to Lime. In order to accurately determine which explanation method is preferable, ground truth explanations are needed.

Defining ground truth explanations may be more difficult for different tasks, and different types of data. Additionally, there may be more than one way to explain a particular observation. Datasets with ground truth explanations must include all possible ways to explain each observation. Failing to include all possible ground truth explanations can unfairly penalize an explanation method for identifying a correct explanation not included in the ground truths.

There is existing work on qualitative evaluation of explanations. Poursabzi-Sangdeh et al. (2021) perform a user experiment to determine what makes a model interpretable. Jeyakumar et al. (2020) perform a user experiment to determine what style of explanation is preferred by users. Adebayo et al. (2020) develop a series of debugging tests, and include a user experiment to determine if users can identify defective models. Not much existing research focuses on quantitatively evaluating all test set explanations for quantitative comparisons across explanation methods. Relying on users to evaluate each explanation in the test set does not scale to large datasets, and cannot be performed on certain types of data (tabular data for example users shown an explanation would not know if its an accurate explanation or not). Additionally, users without a background in machine learning may not be able to determine a good explanation. For quantitative evaluations of explanations that scales to large datasets, scoring metrics must be defined that give an accurate representation of the explanation method's performance. Scoring metrics that measure explanation quality can be formally defined with ground truth explanations.

6 Conclusion

In this work, we propose a method to expand Prototype networks to identify relevant features in the input space. We compare selected examples against a state-of-the-art explanation method on tabular data and verify that the explanations are similar. On image data however, our approach passes the model parameter randomization test but fails the data randomization test. It is common practice in the field of XAI to compare explanation methods using a few selected examples. This is not a thorough evaluation of explanation quality.

We discuss the development of explanation methods, identifying when one explanation method is preferable to another, and the complications that arise when measuring explanation quality. Much research in the field of XAI is devoted to developing new explanation methods. This paper points out that more work should be devoted to evaluating the quality of explanation generated. Many of these issues can be solved with ground truth explanations. We recognize this can be difficult with tabular data. Research should be devoted to defining ground truth explanations for all domains in order to quantitatively evaluate explanations.

References

Adebayo, J.; Gilmer, J.; Muelly, M.; Goodfellow, I. J.; Hardt, M.; and Kim, B. 2018. Sanity Checks for Saliency Maps. In Bengio, S.; Wallach, H. M.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 31*.

Adebayo, J.; Muelly, M.; Liccardi, I.; and Kim, B. 2020. Debugging Tests for Model Explanations. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*. Curran Associates, Inc.

Bach, S.; Binder, A.; Montavon, G.; Klauschen, F.; Müller, K.-R.; and Samek, W. 2015. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLOS ONE*.

Chen, C.; Li, O.; Tao, D.; Barnett, A.; Rudin, C.; and Su, J. 2019. This Looks Like That: Deep Learning for Interpretable Image Recognition. In Wallach, H. M.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E. B.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 32*.

Jeyakumar, J. V.; Noor, J.; Cheng, Y.; Garcia, L.; and Srivastava, M. B. 2020. How Can I Explain This to You? An Empirical Study of Deep Neural Network Explanation Methods. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems 33*.

Kim, B.; Wattenberg, M.; Gilmer, J.; Cai, C. J.; Wexler, J.; Viégas, F. B.; and Sayres, R. 2018. Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV). In Dy, J. G.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden*, Proceedings of Machine Learning Research. PMLR.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the Institute of Radio Engineers*.

Li, O.; Liu, H.; Chen, C.; and Rudin, C. 2018. Deep Learning for Case-Based Reasoning Through Prototypes: A Neural Network That Explains Its Predictions. In McIlraith, S. A.; and Weinberger, K. Q., eds., *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press.

Lundberg, S. M.; and Lee, S. 2017. A Unified Approach to Interpreting Model Predictions. In Guyon, I.; von Luxburg, U.; Bengio, S.; Wallach, H. M.; Fergus, R.; Vishwanathan, S. V. N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30*.

Ming, Y.; Xu, P.; Qu, H.; and Ren, L. 2019. Interpretable and Steerable Sequence Learning via Prototypes. In Teredesai, A.; Kumar, V.; Li, Y.; Rosales, R.; Terzi, E.; and Karypis, G., eds., *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019*. ACM.

Montavon, G.; Lapuschkin, S.; Binder, A.; Samek, W.; and Müller, K. 2017. Explaining nonlinear classification decisions with deep Taylor decomposition. *Pattern Recognit.*

Pace, R. K.; and Barry, R. 1997. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33: 291.

Poursabzi-Sangdeh, F.; Goldstein, D. G.; Hofman, J. M.; Vaughan, J. W.; and Wallach, H. M. 2021. Manipulating and Measuring Model Interpretability. In Kitamura, Y.; Quigley, A.; Isbister, K.; Igarashi, T.; Bjørn, P.; and Drucker, S. M., eds., *CHI '21: CHI Conference on Human Factors in Computing Systems*. ACM.

Precup, D.; and Teh, Y. W., eds. 2017. *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia*, Proceedings of Machine Learning Research. PMLR.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In Krishnapuram, B.; Shah, M.; Smola, A. J.; Aggarwal, C. C.; Shen, D.; and Rastogi, R., eds., *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM.

Selvaraju, R. R.; Das, A.; Vedantam, R.; Cogswell, M.; Parikh, D.; and Batra, D. 2016. Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization. *CoRR*, abs/1610.02391.

Shrikumar, A.; Greenside, P.; and Kundaje, A. 2017. Learning Important Features Through Propagating Activation Differences. In (Precup and Teh 2017).

Shrikumar, A.; Greenside, P.; Shcherbina, A.; and Kundaje, A. 2016. Not Just a Black Box: Learning Important Features Through Propagating Activation Differences. *CoRR*, abs/1605.01713.

Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2014. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In Bengio, Y.; and LeCun, Y., eds., *2nd International Conference on Learning Representations, ICLR 2014, Workshop Track Proceedings*.

Smilkov, D.; Thorat, N.; Kim, B.; Viégas, F. B.; and Wattenberg, M. 2017. SmoothGrad: removing noise by adding noise. *CoRR*, abs/1706.03825.

Springenberg, J. T.; Dosovitskiy, A.; Brox, T.; and Riedmiller, M. A. 2015. Striving for Simplicity: The All Convolutional Net. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, Workshop Track Proceedings*.

Sundararajan, M.; Taly, A.; and Yan, Q. 2017. Axiomatic Attribution for Deep Networks. In (Precup and Teh 2017).

Wachter, S.; Mittelstadt, B. D.; and Russell, C. 2017. Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR. *CoRR*, abs/1711.00399.

Zeiler, M. D.; and Fergus, R. 2014. Visualizing and Understanding Convolutional Networks. In Fleet, D. J.; Pajdla, T.; Schiele, B.; and Tuytelaars, T., eds., *Computer Vision - ECCV 2014 - 13th European Conference*, Lecture Notes in Computer Science. Springer. ISBN 978-3-319-10589-5.