Do Explanations Explain? Model Knows Best

Ashkan Khakzar[1*], Pedram Khorsandi[1*], Rozhin Nobahari[2*], Nassir Navab[1]
ashkan.khakzar@tum.de

[1] Technical University of Munich, Germany
[2] Mila, Quebec Artificial Intelligence Institute, Canada

Abstract

It is a mystery which input features contribute to a neural network's output. Various explanation (feature attribution) methods are proposed in the literature to shed light on the problem. One peculiar observation is that these explanations (attributions) point to different features as being important. The phenomenon raises the question, which explanation to trust? We propose a framework for evaluating the explanations using the neural network model itself. The framework leverages the network to generate input features that impose a particular behavior on the output. Using the generated features, we devise controlled experimental setups to evaluate whether an explanation method conforms to an axiom. Thus we propose an empirical framework for axiomatic evaluation of explanation methods. We evaluate well-known and promising explanation solutions using the proposed framework. The framework provides a toolset to reveal properties and drawbacks within existing and future explanation solutions.[1]

1. Introduction

Considering a neural network function, how do we know which features (patterns) within the input are important for its output? The problem is called feature attribution [16, 35], and the solutions are commonly known as explanation, attribution, or saliency methods. There is an extensive list of explanation methods in the literature [8,9,13,15,16,19,26,27,29,32,33,35,39,41]. One peculiar observation is that these solutions point to different features as being important. Though they are solutions to the same problem, feature attribution, the resulting explanations are curiously dissimilar. The phenomenon raises the question, which explanation is correct? Or are the explanations correct but revealing the problem in a different light?

One approach is to compare the explanations against ground truth (e.g., bounding box) annotations on the dataset [27,38,41]. But how do we know what is important for a human is also important for the model? There is no guarantee (or reason) that the model would use the same features as humans. To resolve this issue, we need to take a step back and ask what it means for a feature to be "important" for an output. The intuitive approach is to remove the feature and observe the output behavior [8, 11, 25]. Such removal of evidence is indeed the foundation of many explanation approaches [8, 9, 16, 26, 35]. However, such a conception could lead to ambiguities. Consider the scenario of having equivalent features (e.g., repeated features), where the existence of each feature alone suffices for a specific output value. Add to the scenario that the removal of any of these features does not affect the output value. In this case, the conception based on removal assigns zero importance to each feature. However, a desirable property, in this case, could be assigning equal importance to each feature.

The concept of importance can thus be further chiseled by specifying *desirable properties* that an importance assignment method ought to satisfy. Such *desirable properties are formalized via axioms* [16, 34, 35]. The axiomatic view provides a complementary framework for evaluating feature attribution solutions. Explanation methods can be evaluated whether they conform to an axiom. The axiomatic view has the advantage that the methods can be mathematically proven to comply with a particular axiom. For instance, solutions such as the Shapley value [16, 28] and integrated gradients [34, 35] are proven to conform with particular axioms. However, proofs can be broken in practical implementations. For instance, [34] show that inherent assumptions within methods that approximate the Shapley value result in methods not conforming with the axioms. Moreover, certain conditions might be overlooked in proofs. Thus *experiments are required to test whether final solutions comply with the axioms*. Even if methods are accompanied by elegant and solid mathematical derivations and proofs, they must comply with the axioms in observations

*denotes equal contribution
[1] https://github.com/CAMP-eXplain-AI/Do-Explanations-Explain

in designed experiments. If they do not comply with axioms in experiments, we may revisit our assumptions and methodologies. Such is the way of the scientific method.

This work lays out an experimental framework for evaluating attribution solutions axiomatically. We set up each experiment such that the solution can be tested whether it complies with a specific axiom. We generate input features that impose a particular behavior on the network's input/output relationship. Features are generated via optimization on the input space while the network parameters are kept constant. Using optimization, we can impose the desired relationship between the generated input and the output. We can thus engineer setups to evaluate axioms. For instance, one axiom that attribution methods are required to conform to is the Null-player axiom. The null-player axiom requires the following; If removal of a feature in all possible coalitions with other features does not affect the output, it should be assigned zero importance. With our proposed framework, we can generate a null player feature for the neural network function. Subsequently, we can test different feature attributions solutions and check whether they assign importance to the null player feature. Thus we can test whether a solution conforms to the Null-player axiom. We also devise experiments to evaluate the explanations in terms of other desirable properties; The class-sensitivity and the feature-saturation. With our framework, we evaluate well-known and recently introduced promising solutions. With our experiments, we intend to reveal properties and drawbacks within existing explanations.

2. Background and Related Work

2.1. Background

We first introduce the feature attribution literature as our framework is designed to evaluate these methods. Then we introduce feature visualization/generation methods since they can be used within our framework.

2.1.1 Explaining Predictions via Feature Attribution

The feature attribution problem is concerned with identifying the input features that contribute to the output value. The solutions can be roughly categorized as follows (some solutions belong to multiple categories).

Backpropagation [4, 30] linearly approximate the network and propose the gradient as attribution. Deconvolution [37], GuidedBackProp [32] backpropagate a modified gradient. Integrated Gradients [35] distributes the change in output with respect to a baseline input by integrating gradients between the two input states. LRP [19], DeepLIFT [29] bacpropagate contribution layer-wise. The contribution notion in LRP and DeepLIFT is also grounded on *removal*.

Perturbation/Removal Methods in this category are explicitly grounded on the removal of features. They mask/perturb input features and observe the output change [8, 9, 18, 23]. E.g., Extremal Perturbations [8] searches for the smallest region in the input such that the keep the region preserves the target prediction. [37] propose occluding pixels or a patch of pixels and measure the output change. IBA [26] inserts an information bottleneck by removing hidden features (via replacing them with noise) and keeps the smallest region that preserves the predictive information. InputIBA [39, 40] enables inserting the information bottleneck on the input.

Latent Features CAM/GradCAM [27, 41] leverage activation values (aka network's attention) of convolutional layers. GradCAM++ uses different summation rules on layers and is applicable to all layers. IBA [26] also utilizes latent features. FullGrad leverages the activation, gradient and, bias values from all layers. PathwayGrad [13] leverages critical pathways (pathway important neurons).

Game Theory The attribution problem can be considered as credit assignment in cooperative game theory. This is achieved by presuming the network's function is a score function, and input features are players. A solution to this problem that satisfies several axioms is the Shapley Value. This notion is also grounded upon the removal of players and the effect of removal on score function. Shapley Value considers the removal of a player in all possible coalitions. Due to computational complexity, several approximations are proposed for neural networks. DeepSHAP [16] backpropagates SHAP values via DeepLift [29] framework. It is recently shown [34] that Integrated Gradients [35] approximates Shapley value in continuous setting.

2.1.2 Generating Features that Activate a Neuron

These works identify what input patterns/features activate a neuron and are commonly referred to as feature visualization. In essence, the methods generate images that maximize certain neuron activations [7, 17, 20, 22, 30, 36]. This can be achieved by performing the optimization on the image while freezing networks parameters. We can use any of these solutions within our framework. We opt for deep image prior [36] (refer to Sec. 3.5). Another method that we utilize within our framework is the adversarial patch [5].

2.2. Related Work

This section introduces the works that evaluate explanations. Our framework belongs to both ”ground truth” and ”axiomatic” categories. We discuss the differences to existing works in each section separately (more in appendix).

Do Explanations Explain? An early work [21] demonstrates that Deconvolution [37] and Guided Backpropagation [32] are reconstructing image features rather than explaining the prediction. Thus an explanation can be visually interpretable but not really be an explanation. The works in this section investigate whether an explanation method is indeed explaining the prediction and whether it can be trusted. Each work evaluates an explanation from a different vantage point. We categorize the works as follows:

Perturbation/Removal The objective of these works is to evaluate whether features identified as salient by attribution methods are indeed contributing to the output. The intuition behind them is that if the identified features are important, perturbing (removing) them changes the output relatively more. Sensitivity-N [2] and [25] use various perturbation schemes on the input and observe the output change. Remove-and-Retrain [11] perturbs the input, then retrains the model and measures the accuracy drop.

Ground Truth These works compare the explanations with a ground truth of features that are important. Pointing game [38] and classic localization-based metrics [27] use annotations on natural images done by humans. However, here there is an underlying assumption that the model is using the same features as humans, which is a crude assumption. To solve this issue, CLEVR XAI [3] proposes generating a synthetic dataset using CLEVR [12]. The model is then trained on the generated dataset, and then explanations are compared with the ground truth. The approach adds partial control. However, there is no guarantee that the model picks up the intended features in the generated dataset. In our framework we can control what features contribute or do not contribute to the model's output.

Axiomatic Axiomatic approaches check whether the model complies with particularly desirable properties. Evaluation can be either theoretical, where the method is proven to satisfy an axiom (or a desirable property), [16, 31, 34, 35], or experimental. Sanity checks [1] experimentally check whether randomizing network parameters change the explanation. Another desirable property is class sensitivity, i.e., the explanation should not be the same for different outputs (classes) if their contributing features differ. [14] provides a reasoning on why several methods are insensitive to parameter randomization and different classes. [21, 24] propose experiments to evaluate class-sensitivity on natural image datasets. However, on natural images, there is no guarantee that the model uses different features for different classes. Our framework provides a controlled setup as features are generated.

3. Methodology

The objective is to have a controlled experimental environment in which we control what features contribute or do not contribute to the output of the neural network function. In this environment, we can devise scenarios to test the explanations against axioms. To this end, we leverage the model itself and run an optimization on the input, thus controlling how features contribute to the output.

Importance or contribution is understood only with respect to a reference/baseline state (”removal” is setting feature values to a reference value). In our setup, we compute the contribution of a feature with respect to a reference of normal random noise. X denotes the reference input. We refer to a group of pixels and their specific values as a ”feature”. In this work, we select a patch of pixels to form the feature. We denote the patch/feature by f and a baseline input that has a feature f added to it by $X_{\{f\}}$ and the neural network function for a target output t by $\Phi_t(.)$. To generate the feature f that corresponds to a target t we generate a patch on the baseline input X that activates the target t. Since the added feature changes the output value (by design), according to sensitivity axiom [35] it is guaranteed that it contributes to the output. The optimization is performed only on patch f (not on other areas of input X). The optimization loss for generating feature f corresponding to target t is denoted by $\mathcal{L}^t_f$. Depending on the scenario, $\mathcal{L}^t_f$ can be associated with either of the following. We can either generate a patch that maximizes the target value, $\min_f \; -\Phi_t(X_{\{f\}})$ or generate a patch that achieves a constant target value c, $\min_f \; \mathcal{L}_{CE}(\Phi_t(X_{\{f\}}), c)$ where $\mathcal{L}_{CE}$ denotes cross-entropy loss.

3.1. Null Feature

The objective in this section is to devise a setup for testing the null feature axiom. A null feature is one that does not contribute to the output score. If a feature is a null feature, it is a desirable property for the explanation not to assign any contribution to that feature. based on cooperative game theory and attribution literature, null feature can be formally defined as follows. Having a group of features (players), a feature is a null feature if its absence does not affect the output score function in all possible coalitions of features. I.e. if we have a set of n features $\{f_1, ..., f_n\}$, a feature f_i is null for output $\Phi_t(.)$ if $\Phi_t(X_{\{f_i \cup \mathcal{S}\}}) = \Phi_t(X_{\{\mathcal{S}\}})$, where $\mathcal{S}$ denotes all subsets of features excluding f_i, i.e. $\mathcal{S} \subset \{f_1, ..., f_n\} \setminus \{f_i\}$. Note that are 2^{N-1} possible coalitions.

In our experimental setup, we add two features to the baseline input X (one can add more features and devise more complex or creative experiments). We add feature f_a that corresponds to output $\Phi_a(.)$ and add another feature f_{null} that corresponds to an output $\Phi_b(.)$ but is a null feature for output $\Phi_a(.)$. In order for the f_{null} to be a null feature, its absence in all possible coalitions with f_a should have no effect on output $\Phi_a(.)$. The are two possible coalitions, which are the subsets of f_a, namely f_a and . Therefore, the output $\Phi_a(.)$ must stay constant when f_{null} is removed when f_a exists in baseline input X. The output $\Phi_a(.)$ must also stay constant when f_{null} is added to the baseline X. Therefore, the optimization problem is defined as the following two concurrent optimizations,

$$\min_{f_a} \mathcal{L}^a_{f_a} \tag{1}$$

$$\begin{aligned}\min_{f_{null}} \mathcal{L}^b_{f_{null}} &+ (\Phi_a(X_{\{f_a,f_{null}\}}) - \Phi_a(X_{\{f_a\}}))^2 \\ &+(\Phi_a(X_{\{f_{null}\}}) - \Phi_a(X_{\{\}}))^2\end{aligned} \tag{2}$$

where $\mathcal{L}^a_{f_a}$ generates feature f_a corresponding to output $\Phi_a(.)$. In Eq. (2) $\mathcal{L}^b_{f_{null}}$ generates a feature f_{null} corresponding to output $\Phi_b(.)$. The second and third term in Eq. (2) try to make f_{null} be a null feature for $\Phi_a(.)$ by removing it in possible coalitions with f_a. The result of the optimization is $X_{\{f_a,f_{null}\}}$, which is a baseline noise image X that contains patches/features f_a and f_{null}. In this setup, we aim to test whether an explanation method attributes the output $\Phi_a(X_{\{f_a,f_{null}\}})$ to the null feature. The proposed metric for evaluation is provided in Sec. 3.4

3.2. Class Sensitivity

Another property that is expected from an explanation method is class sensitivity, i.e. output sensitivity. Considering two outputs $\Phi_a(.)$ and $\Phi_b(.)$ of a neural network, if the contributing input features to the these outputs differ, the explanations for the outputs should also be different. To test such property we devise two scenarios:

3.2.1 Single Feature Scenario

In our first proposed setup, we only add one feature f_a to the reference input X. The feature is generated such that it corresponds to the output $\Phi_a(.)$ but is a null feature for another output $\Phi_a(.)$. Therefore,

$$\min_{f_a} \mathcal{L}^a_{f_a} + (\Phi_b(X_{\{f_a\}}) - \Phi_b(X_{\{\}}))^2 \tag{3}$$

where the first term $\mathcal{L}^a_{f_a}$ generates the patch f_a on reference input X, and the second term makes sure it is a null feature for output $\Phi_b(.)$. I.e. the removal of feature f_a should not affect the output $\Phi_b(.)$.

In this setup, the explanations for the two outputs $\Phi_a(.)$ and $\Phi_b(.)$ are compared. It is expected that the first explanation (for $\Phi_a(.)$) attributes the output (partly) to f_a. Whereas, the second expalanation (for $\Phi_b(.)$) should not attribute the prediction of $\Phi_b(.)$ to the the feature f_a. Our proposed metric for evaluating this effect is provided in Sec. 3.4.

3.2.2 Double Feature Scenario

In this setup we add two features f_a and f_b to the reference input X, each corresponding to the different outputs $\Phi_a(.)$ and $\Phi_b(.)$ respectively. In this setup the dominantly contributing feature to $\Phi_a(.)$ is feature f_a and the dominantly contributing feature to $\Phi_b(.)$ is f_b. Therefore we perform two concurrent optimizations. The first one,

$$\min_{f_a} \mathcal{L}^a_{f_a} + (\Phi_b(X_{\{f_a,f_b\}}) - \Phi_b(X_{\{f_b\}}))^2 \tag{4}$$

generates f_a which contributes to output $\Phi_a(.)$ but its removal in the presence of feature f_b does not affect output $\Phi_b(.)$. The second optimization,

$$\min_{f_b} \mathcal{L}^b_{f_b} + (\Phi_a(X_{\{f_a,f_b\}}) - \Phi_a(X_{\{f_a\}}))^2 \tag{5}$$

generates f_b which contributes to output $\Phi_b(.)$ but its removal in the presence of feature f_a does not affect output $\Phi_a(.)$. Thus the dominantly contributing feature for $\Phi_a(.)$ is f_a and for $\Phi_b(.)$ is f_b.

In this scenario we expect the explanations to switch from feature f_a to f_b when the output to be explained is changed from $\Phi_a(.)$ to $\Phi_b(.)$. Our proposed metric for capturing this metric is provided in section Sec. 3.4.

3.3. Feature Saturation

In this section, we devise a scenario where features saturate the output. Such that the features f_{a1} and f_{a2} together (i.e. $X_{\{f_{a1},f_{a2}\}}$) result in the same output value as when the features are added to reference input X individually. To achieve this, we solve two optimizations concurrently,

$$\min_{f_{a1}} \mathcal{L}^a_{f_{a1}} + (\Phi_a(X_{\{f_{a1},f_{a2}\}}) - \Phi_a(X_{\{f_{a2}\}}))^2 \tag{6}$$

where the first term generates f_{a1} such that the output is equal to a constant value c. The second term makes sure that feature f_{a1} removal from input does not affect the output when f_{a2} is present. The second optimization does this procedure on the second feature f_{a2},

$$\min_{f_{a2}} \mathcal{L}^a_{f_{a2}} + (\Phi_a(X_{\{f_{a1},f_{a2}\}}) - \Phi_a(X_{\{f_{a1}\}}))^2 \tag{7}$$

In this setup, the existence of one of the features is sufficient for the prediction. As they contribute equally to the output, an explanation solution is expected to attribute the output equally to both features. Our proposed metric for evaluating this property is provided in Sec. 3.4.

3.4. Metrics

In this section, we introduce our metrics for evaluating the properties in each of the generated setups. We denote an explanation generated for target output $\Phi_t(.)$ by S_t.

Null Feature Metric Defined as contributions assigned to null feature relative to total assigned contributions:

$$\frac{\sum_{f_a} S_a}{\sum_{S_a} S_a} \tag{8}$$

The sum operator $\sum_f S_t$ runs over all corresponding pixels in S_t that are in patch f.

Class Sensitivity Metric In the Double Feature Scenario, we measure the class sensitivity by:

$$\frac{\sum_{f_a \cup f_b} min(S_a,\, S_b)}{\sum_{f_a} S_a + \sum_{f_b} S_b} \tag{9}$$

where the $min(S_a,\, S_b)$ is the pixel-wise minimum of S_a and S_b. In an extreme case, for the explanation method being indifferent towards the target class, the $min(S_a,\, S_b)$ would be equal to S_a and S_b. Therefore, the metric evaluates to one. In the other extreme, where attributions shift from f_a to f_b, the $min(S_a,\, S_b)$ and the metric is zero.

For the Single Feature Scenario, the class sensitivity is:

$$corr(\frac{\overline{S_a - S_a \backslash f_a}}{\overline{S_a \backslash f_a}}, \frac{\overline{S_b - S_b \backslash f_a}}{\overline{S_b \backslash f_a}}) \tag{10}$$

The term $\frac{\overline{S_t - S_t \backslash f}}{\overline{S_t \backslash f}}$ determines the average amount of contribution score inside patch f, devided by the average outside the patch. The higher correlation implies the method is attributing to the same feature for both outputs Φ_a and Φ_b.

Feature Saturation Metric To evaluate how the attribution is distributed between the features, we evaluate the correlation between attributions assigned to feature f_{a1} and f_{a1}

$$corr(\sum_{f_{a1}} S_a, \sum_{f_{a2}} S_a) \tag{11}$$

A method that assigns the attribution to only one feature receives a lower score.

3.5. Implementation Details

Reference/Baseline Input: Importance is understood with respect to a reference state. The reference is chosen such that it represents the missingness of features. In the vision domain, it is customary to use zero value [29, 35], or noise [26]. In any case, our framework is not dependent on the reference. We do not make any assumptions about what features are in the reference. We ensure that a feature is null with respect to the reference, and our metric only considers the generated features and not the background (we use attributions in background only for scaling).

Deep Prior Network: If we perform the optimization on the patches without any regularization, it is easy to get trapped in local solutions. In this case, we may not achieve a satisfactory optimization solution for Eq. 1-7. In our work, we leverage ”deep image prior” [36] to limit the space of solutions and avoid trivial local solutions. Using deep image prior methodology, we add a decoder network with random weights and a random seed input behind the generated patch. In other words, the patch is parameterized by the prior network. The optimizations are thus done on the parameters of the prior network instead of the patch. In [36] it is also demonstrated that the untrained network does capture some of the low-level statistics of natural images. Therefore the generated patches also look interpretable to us. Visual interpretability is not required within this framework, though it can make the experiments more intuitive.

The Model: The choice of the model does not affect the framework as long as the optimizations are solvable. The network is pre-trained on ImageNet [6]. However, the proposed framework does not depend on the network being trained. For a random network the generated features would not ”look” interpretable.

Optimization During optimization steps, we place the patch in different locations to ensure that the results do not depend on the patch’s location. Moreover, in order to balance the terms in Eq. 1-7 we use focal loss [10] (appendix).

4. Results and Discussion

The objective of our proposed framework is to reveal insights and shortcomings regarding explanation methods. We evaluate various explanation methods from different categories. DeepSHAP and IntegratedGradient are theoretically axiomatic methods. GradCAM and GradCAM++ are two popular methods that leverage network attention. We also evaluate the recently introduced FullGrad from this family. In addition, we evaluate two recent promising solutions, IBA and Extremal Perturbations.

4.1. Null Feature

The null feature experiment checks whether an explanation attributes the output to a null feature. I.e., it checks whether the explanation method identifies the null feature as important. The framework guarantees that the null feature is not contributing. Using the framework, we generate 1000 inputs. For each input sample, the feature is generated for a random output. We then proceed and compute the null feature metric on each generated inputs and report the average in Tab. 1. An example generated input is in Fig. 1.

FullGrad, DeepSHAP, Gradient, and GuidedBackProp perform the worst in this experiment. This performance may point to the fact that these methods identify all features within the input as important. It is previously shown [21] that GuidedBackProp reconstructs image features rather than explaining the prediction, and our results are aligned

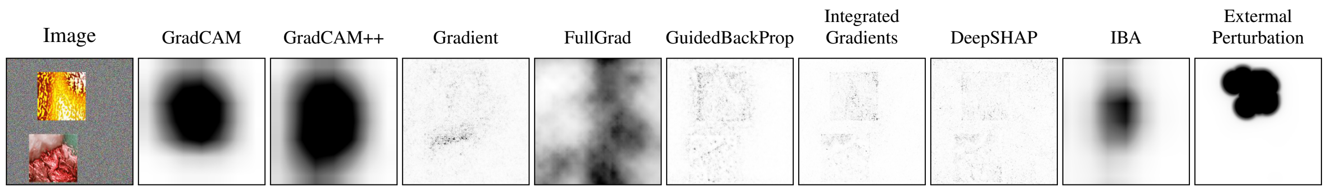

Figure 1. **Null Feature Experiment**: The image on the left represents the generated features on the reference (noise) input. The features are generated using the model itself. Within the image, the lower feature (patch) is generated such that it is a null feature for the output. The rest of the images represent different explanations. As the second feature is a null feature, an explanation method should not assign importance to it. We observe that GradCAM, IBA, and Extremal Perturbation perform best in avoiding the null feature.

Method	Null Feature	Class Sensitivity: Double Feature Scenario	Class Sensitivity: Single Feature Scenario	Feature Saturation
GradCAM	0.135	0.176	0.050	0.243
GradCAM++	0.452	0.469	0.845	-0.571
Gradient	0.835	0.469	0.684	0.310
FullGrad	1.00	0.931	0.951	-0.130
GuidedBackProp	0.704	0.555	0.979	0.703
IntegretedGradient	0.534	0.344	0.759	0.212
DeepSHAP	1.03	0.507	0.934	0.221
IBA	0.211	0.191	0.295	-0.223
ExtermalPerturbation	0.047	0.039	0.759	-0.680

Table 1. Evaluation of Explanations with the Framework: **1) Null Feature**: Null feature experiment evaluates the extent to which each explanation attributes the output to a null feature. In this metric, the less the value, the better. Extremal Perturbation [], GradCAM, and IBA are the favorable methods from this null feature perspective. **2) Class Sensitivity**: For both experiments, the lower the value, the better 1) Double Feature Scenario: In the case where two features corresponding to two different classes are present, Extremal Perturbation, IBA, and GradCAM attribute to the correct feature when applied to the two outputs. 2) Single Feature Scenario: In the case where only feature is present, explanations for two different outputs are similar to all methods except GradCAM and, to some extent, IBA. **3) Feature Saturation**: the experiment evaluates how explanations distribute the importance between saturated features. In this metric, the higher the value, the better. The notable observation is Extremal Perturbation, as it identifies only one of the features as important.

with the finding. It can also be inferred that gradient is also sensitive to all features in the input. DeepSHAP is widely known as a solid method as it involves SHAP. However, it also has a backpropagation mechanism (as it is engineered on DeepLift). It seems the backpropagation is the culprit, as other gradient methods also fail this experiment. FullGrad does a weighted sum of gradients and biases of all layers. The gradients in the early layers can be the culprit in this case. We observe that GradCAM rarely assigns attribution to the null feature. And the assigned values may be due to CAM's low resolution. IBA and extremal perturbation are both grounded on the removal of features. We see that they also avoid attributing to the null feature.

We also evaluate IBA and GradCAM++ on different layers of a ResNet network. Among the advantages of these methods is that they can be applied to early layers to produce higher resolution maps. However, we observe in Fig. 2 and Tab. 2 that as we move towards early layers, the methods attribute to the null feature.

4.2. Class Sensitivity

Double Feature Scenario The objective is to observe how the explanations for two different outputs differ when both outputs have corresponding features present. The metric results are presented in Tab. 1, and visual examples are presented in Fig. 3. We observe that GradCAM, IBA, and Extremal Perturbation attribute the corresponding features when explaining the different outputs. FullGrad produces the same explanation when applied to the two outputs. We observe that Gradient, GuidedBackProp, DeepSHAP, and IntegratedGradient slightly switch explanations. We also perform layerwise experiments for IBA and GradCAM++. We observe that the explanations become less class sensitive in earlier layers.

Single Feature Scenario In this setting, we evaluate class sensitivity in the case where only one contributing feature is available. Suppose the explanation for the output of the

	IBA				GradCAM++			
Metric	layer 1	layer 2	layer 3	layer 4	layer 1	layer 2	layer 3	layer 4
Null Feature	0.315	0.311	0.201	0.211	0.827	0.906	0.815	0.453
Double Feature Scenario	0.327	0.337	0.207	0.191	0.977	0.948	0.899	0.469
Single Feature Scenario	0.219	0.237	0.158	0.295	0.979	0.823	0.761	0.845

Table 2. Evaluations of IBA and GradCAM++ explanations for various layers of ResNet: IBA and GradCAM++ are applicable to different layers of convolutional networks. However, we observe that as we move towards earlier layers (toward the input), more attribution is assigned to the null feature. We also observe the same trend with class sensitivity. The results significantly deteriorate for GradCAM++ (in both experiments, the lower the value, the better). It is thus advisable to apply these explanations to deep layers.

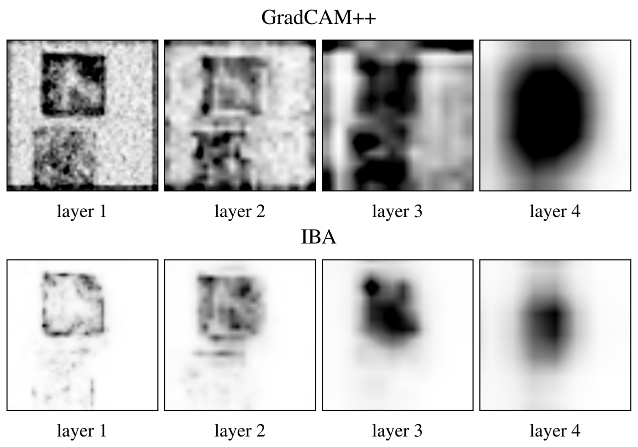

Figure 2. Null Feature Experiment for IBA and GradCAM++ on different layers of a network. The second (lower) feature is a null feature. We observe that as we move toward earlier layers, the explanations attribute to the null feature for both methods.

corresponding feature is similar to the explanation for an output to which the feature does not contribute. In that case, the explanation is not sensitive to the output. A visual example for this case is provided in Fig. 4. The results for the associated metric are provided in Tab. 1. In this scenario, more explanation methods are prone to attribute the output to the single feature within the image. Interestingly the only method that is sensitive to output, in this case, is GradCAM. Even IBA and Extremal perturbation that performed well in double feature scenario identify the same feature for the two outputs. This might be the property for all perturbation/removal-based methods, that they converge to the only predictive feature within the input, even if the feature is predictive for another class.

4.3. Feature Saturation

This experiment aims to check how an attribution method behaves in case there are saturated features present in the input. The desirable property, in this case, is to attribute to both features. The results of the metric are provided in Tab. 1. A visual example is presented in the appendix (Fig. 5). The two features (patches) in the input contribute equally to the output, and the presence of only one is enough for the exact output prediction. We expect to observe that a method such as Extremal Perturbation attributing to only one of the features. The method searches for the smallest region that keeping it would keep the output prediction. In the case of saturated features, this translates to keeping only one feature. The metric in Tab. 1 shows that statistically, the method converges to one of the features. The other methods are mostly attributing to both features, but considering the results from the Null player experiment and the Class Sensitivity experiment, the observation better be interpreted with caution. Several methods might be attributing to both features because they attribute to all features for other reasons. For instance, we observed with GuidedBackprop that the method is attributing to null feature and is attributing to both features when explaining different outputs. A method that is attributing to both features, but does that in null feature case as well, is not doing the attribution for fair distribution, but for other reasons.

4.4. Discussion on the Framework

Is the optimization feasible? To practically guarantee that the setups (Eq. 1-7) are realized, we set stopping criteria and continue the optimization until the desired setup is achieved. The stopping criteria checks if the properties (e.g., one feature being null) are satisfied within a certain threshold. The setups are variations of removing patches; we thus check the effect of removing patches on output in all setups after each epoch. We report the output change ratio (output-change / output) when removing patches in different setups (null, class-sensitivity, saturation) for an average of 1K samples. For the Null Feature, Feature Saturation, and Class-Sensitivity (double), the ratio is 0.0712. For Class-sensitivity (single) ratio is 0.0649.

Is the framework sensitive to how the samples are generated? By definition, given a function, for any input/output, the attribution method ought to identify the contribution of features to the output. The definition is for *any* input and is regardless of the generated input's properties (e.g. being out of distribution). The framework makes sure

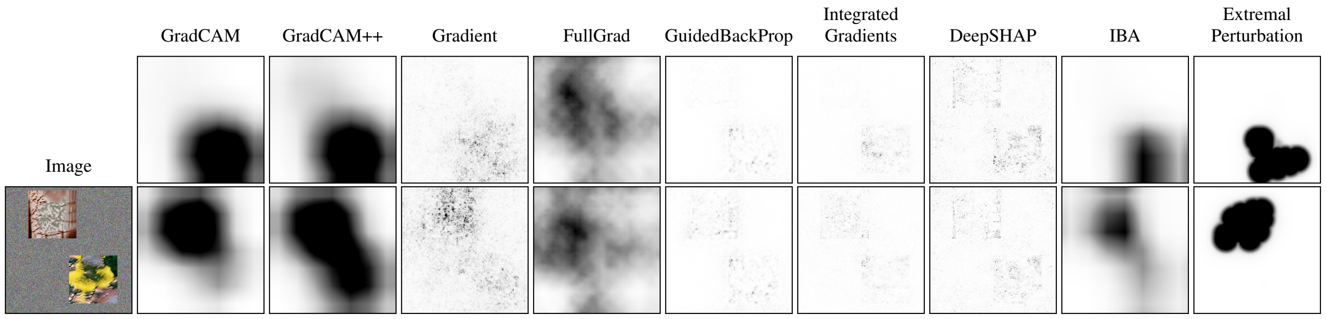

Figure 3. **Class Sensitivity - Double Feature Scenario**: The image on the left represents the generated features on the reference input. The features are generated such that each corresponds to a different output. The lower feature (patch) corresponds to the first output (first row), and the other patch corresponds to the second output (second row). It is expected that explanations for the two outputs differ and attribute to the corresponding feature of each output. GradCAM, IBA, and Extremal Perturbation manifest this property.

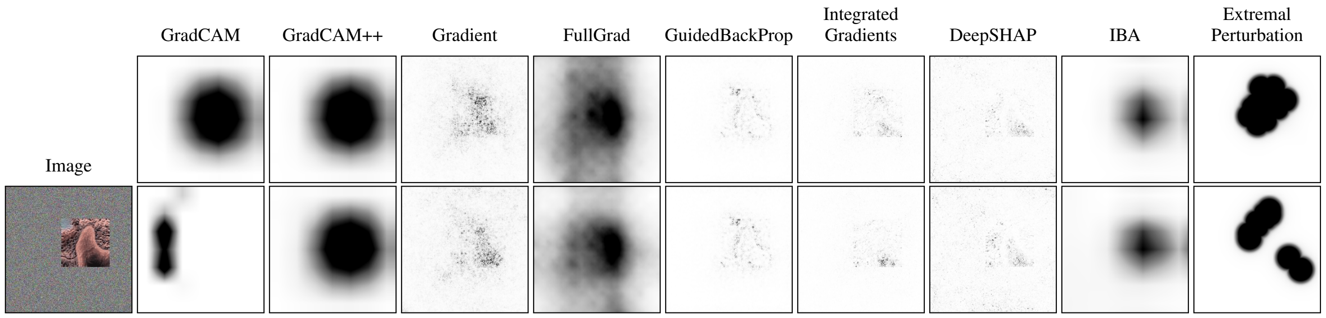

Figure 4. **Class Sensitivity - Single Feature Scenario**: The feature is generated such that it contributes to one output and is null for another output. The first output is presented in the first row. The output to which the feature is null is presented below. The explanations are presented for both outputs. It is expected that the explanations for the two outputs differ. Moreover, the explanations should not attribute to the feature for the null output (second row.). The only method that attributes correctly, in this case, is GradCAM.

that the feature has a specific behavior, e.g., having zero contribution to the output.

Novel insights from the framework in a nutshell: We reveal FullGrad, GradCAM++, Integrated Gradients and Gradient are attributing to null feature. We practically affirm DeepSHAP breaks axioms (theory in [34]). We show CAM, Extremal Perturbations (Exp), and IBA as trustworthy in terms of null and class-sensitivity axiom (though when only one feature is present, only CAM prevails). We reveal saturation properties within ExP and IBA. We reveal GradCAM++, FullGrad, Gradient, IG, DeepSHAP can be class-insensitive. We show (Tab. 2) IBA and GradCAM++ break axioms in early layers (though they were proposed to work on other layers than the deepest).

5. Conclusion

This work proposes an experimental framework for axiomatic evaluation of explanation methods using the model. Within the framework, the explanations are checked whether they comply with an axiom or satisfy a property. The experimental setup is realized through generating features using the model. Through feature generation, several scenarios for evaluating axioms are introduced. The framework reveals that many explanation methods identify a null feature as salient, even though the framework guarantees the feature to have no contribution. Moreover, the framework shows many explanations are not class sensitive and generate roughly equivalent explanations for different outputs. The only methods that do not attribute to null features and are class sensitive are GradCAM, IBA, and Extremal Perturbations. We further analyze IBA and GradCAM++ on various layers of a neural network and reveal that the axioms are complied with only if they are applied to the final layer. Our proposed framework can be used to evaluate upcoming explanation methods. Furthermore, researchers can add more creative experiments to the proposed framework to assess explanations from other perspectives.

Acknowledgement The work is supported by Munich Center for Machine Learning (MCML) with funding from the Bundesministerium fur Bildung und Forschung (BMBF) under the project 01IS18036B.

References

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, 2018. 3

[2] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. *ICLR*, 2018. 3

[3] Leila Arras, Ahmed Osman, and Wojciech Samek. Ground truth evaluation of neural network explanations with clevr-xai. *arXiv preprint arXiv:2003.07258*, 2020. 3

[4] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert MÃžller. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11(Jun):1803–1831, 2010. 2

[5] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017. 2

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 5

[7] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009. 2

[8] Ruth Fong, Mandela Patrick, and Andrea Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *ICCV*, 2019. 1, 2

[9] Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *ICCV*, pages 3429–3437, 2017. 1, 2

[10] Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. Dynamic task prioritization for multitask learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 270–287, 2018. 5, 11

[11] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 9737–9748, 2019. 1, 3, 11

[12] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910, 2017. 3

[13] Ashkan Khakzar, Soroosh Baselizadeh, Saurabh Khanduja, Christian Rupprecht, Seong Tae Kim, and Nassir Navab. Neural response interpretation through the lens of critical pathways. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13528–13538, June 2021. 1, 2

[14] Ashkan Khakzar, Soroosh Baselizadeh, and Nassir Navab. Rethinking positive aggregation and propagation of gradients in gradient-based saliency methods. *arXiv preprint arXiv:2012.00362*, 2020. 3, 11

[15] Ashkan Khakzar, Yang Zhang, Wejdene Mansour, Yuezhi Cai, Yawei Li, Yucheng Zhang, Seong Tae Kim, and Nassir Navab. Explaining covid-19 and thoracic pathology model predictions by identifying informative input features. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 391–401. Springer, 2021. 1

[16] Scott M. Lundberg and Su In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, 2017. 1, 2, 3, 11

[17] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015. 2

[18] Saumitra Mishra, Bob L Sturm, and Simon Dixon. Local interpretable model-agnostic explanations for music content analysis. In *ISMIR*, pages 537–543, 2017. 2

[19] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus Robert Müller. Explaining nonlinear classification decisions with deep Taylor decomposition. *Pattern Recognition*, 2017. 1, 2

[20] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in neural information processing systems*, pages 3387–3395, 2016. 2

[21] Weili Nie, Yang Zhang, and Ankit Patel. A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. *International Conference on Machine Learning*, 2018. 3, 5

[22] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization. 2

[23] Zhongang Qi, Saeed Khorram, and Fuxin Li. Visualizing deep networks by optimizing with integrated gradients. *CVPR Workshop*, 2019. 2

[24] Sylvestre-Alvise Rebuffi, Ruth Fong, Xu Ji, and Andrea Vedaldi. There and Back Again: Revisiting Backpropagation Saliency Methods. apr 2020. 3

[25] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*, 2017. 1, 3, 11

[26] Karl Schulz, Leon Sixt, Federico Tombari, and Tim Landgraf. Restricting the flow: Information bottlenecks for attribution. In *International Conference on Learning Representations*, 2020. 1, 2, 5

[27] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017. 1, 2, 3

[28] Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953. 1

[29] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *34th International Conference on Machine Learning, ICML*, 2017. 1, 2, 5

[30] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013. 2

[31] Leon Sixt, Maximilian Granz, and Tim Landgraf. When explanations lie: Why many modified bp attributions fail. *International Conference on Machine Learning*, pages 9046–9057, 2020. 3, 11

[32] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. In *3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings*, 2015. 1, 2, 3

[33] Suraj Srinivas and Francois Fleuret. Full-gradient representation for neural network visualization. Technical report, 2019. 1

[34] Mukund Sundararajan and Amir Najmi. The many shapley values for model explanation. *37th International Conference on Machine Learning, ICML*, 2020. 1, 2, 3, 8, 11

[35] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *34th International Conference on Machine Learning, ICML 2017*, 2017. 1, 2, 3, 5, 11

[36] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018. 2, 5

[37] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014. 2, 3

[38] Jianming Zhang, Sarah Adel Bargal, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. *International Journal of Computer Vision*, 126(10):1084–1102, 2018. 1, 3, 11

[39] Yang Zhang, Ashkan Khakzar, Yawei Li, Azade Farshad, Seong Tae Kim, and Nassir Navab. Fine-grained neural network explanation by identifying input features with predictive information. *Advances in Neural Information Processing Systems*, 34, 2021. 1, 2

[40] Yang Zhang, Ashkan Khakzar, Yawei Li, Azade Farshad, Seong Tae Kim, and Nassir Navab. Fine-grained neural network explanation by identifying input features with predictive information. *arXiv preprint arXiv:2110.01471*, 2021. 2

[41] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, pages 2921–2929, 2016. 1, 2