T-Grad: A Trustworthy Gradient Explanation for Convolutional Neural Networks

Truong Tran$^{1,2[0009-0005-2467-3175]\star}$, Trung Mai$^{1,2[0009-0007-9947-145X]\star}$,

Duc Le$^{1,2[0009-0006-1574-1743]\star}$, and Bac Le$^{1,2[0000-0002-4306-6945]\star\star}$

[1] Faculty of Information Technology, University of Science,
Ho Chi Minh City, Viet Nam
[2] Vietnam National University, Ho Chi Minh City, Viet Nam.
Email: nhattruong1009@gmail.com,quytrungm@gmail.com,
ngocducdarkzonezero@gmail.com, lhbac@fit.hcmus.edu.vn

Abstract. In this paper, we introduce T-Grad, a trustworthy AI interpretation method designed to enhance the understanding of features that explain model behavior in image classification tasks. By combining gradients and biases with advanced upgrades to generate saliency maps, T-Grad offers a more accurate interpretation of a model's true behavior. This method has demonstrated its effectiveness through various evaluations on the ImageNet1K dataset, including performance experiments, visual assessments, and execution speed analyses. We believe that T-Grad can significantly contribute to the overall concept of Explainable AI (XAI) in general and improve the reliability and robustness of image classification models specifically.

Keywords: XAI · Trustworthy AI · Saliency map · Gradients · Bias

1 Introduction

Deep learning models have made many important advances in recent decades and their applications are spread across many fields from healthcare to education. Many fields show that with the help of artificial intelligence, complication and execution time are significantly reduced. However, despite their superior performance on many tasks, deep learning models still need close explanations to gain greater credibility on high-stakes tasks.

Currently, the field of XAI in image processing has developed many different methods, we can group them based on their operating principles. Among them, forward propagation-based methods such as SHAP [6], LIME [9] use the change in results when affecting the input sample to provide explanations. On the other hand, some back propagation methods such as Deep Taylor Decomposition [7], Pixel-wise Explanations [5] use the method of redistributing the impact of each

* These authors contributed equally to this work.
** Corresponding author.

neuron on each layer from prediction back up to the input sample for interpretation.

However, each of the above methods faces certain limitations. For example, LIME requires us to train a simple and explainable model based on the behavior of the current model, SHAP requires a huge amount of time to be able to calculate the result value for a given single sample. Deep Taylor Decomposition and Pixel-wise Explanations methods often give the same results as the model for edge detection - an algorithm that does not require machine learning - reducing the significance of the interpretation results. Class Activation Mapping (CAM) image interpretation methods are often used to create explanations for deep learning models. However, the generated heatmap regions essentially come from the output of the final convolution layer, not the input image. More specifically, these methods ignore the importance of bias values. This can cause the explanation method to not truly describe the model's behavior.

This study highlights shortcomings and several components that previous methods have not exploited. At the same time, the study proposes a more comprehensive approach, taking full advantage of all components in the model to highlight the features learned by the model for interpretation purposes. The method proposed in this study not only improves the accuracy of deep learning model interpretation but also broadens the horizons of how to explain complex models. Thereby, it helps increase confidence and understanding of model decisions, contributing to advancement in this field.

In the remainder of this paper, we present a comprehensive exploration of the proposed method and its importance in the field. 2 provides a summary of existing algorithms and their limitations. 3 delves into the analysis of theoretical foundations and proposed methods. 4 is dedicated to presenting and analyzing our experimental results, comparing the performance of our proposed method with established benchmarks. In 5, we visualize the interpretation method to depict the differences from existing studies, In 6, we evaluate the effectiveness of the method and its approach in solving the challenges posed, thereby providing a perspective for future development.

2 Related works

2.1 CAM methods

These methods are based on the work of Zhou et al [18] on a specially designed network. Whereby the activation map is treated as a single feature for a simple linear classification network, therefore, the influence of the activation map can be directly calculated by its corresponding weight to the specified class. The methods can provide fast, highly localized, and discriminative explanations. However, methods based on CAM usually have large interpretation areas due to the effects of upsampling. GradCAM [10] popularized the CAM method by showing how it can be applied on almost any network. It proposes that on any network, the weight of the activation map's impact on class prediction will be the sum of the impact of every neuron on that activation map. There are also a number of other methods [2],[16],[8],[4] that have been proposed but most of these works are changing how we can compute the weight of the activation map's impact or giving an saliency map that can cover the objects.

2.2 Gradient methods

A common idea in gradients field for giving explanation is finding the highest influence features corresponds to the predicted class. The simplest way is to approximate the model by an linear function using first-order Taylor. Different ways, Guided Backpropagation [13] eliminates the values $w < 0$ from the back-propagation process to help the influencing w avoid being disturbed by negative weights. One benefit we can get from it is the fact that this method eliminates the noises or gradients that have a negative effect on the prediction from the model. However, its weakness is this method has been proven to be categorized as an edge detection method. SmoothGrad [12] is another approach to enhance the calculation for gradients used to explain. It uses noises applied to the sample to determine w at $x_0 + \mathcal{N}(\mu, \sigma^2)$ instead of x_0. However, the weakness of this method is that it takes a long time to compute the explanation.

Hence, using only w to describe the features may yield inappropriate results. For example, consider this linear equation $f(x_1, x_2) = -2x_1 + 9x_2$. We always have $\frac{\partial f}{\partial x_1} = -2$ and $\frac{\partial f}{\partial x_2} = 9$. This indicates that even when $x_2 = 0$ which does not contribute anything to the final value of f, it will still be considered a feature with a significant impact on the result. Consequently, some methods, such as Pixel-wise Explanations [5] and DeepLIFT [11], take into account additional factors from the network's input.

2.3 Using bias

In Wang's study [17] on deep learning networks with activation functions that are linear in their defined range, such as ReLU or HardTanh, it was shown that removing bias before or after training significantly deteriorates the model's classification capability. This is understandable because even if the model is a linear function, bias helps make the "classification equation" more flexible, rather than just passing through the origin. Additionally, when formulating a linear model with bias, it shows that $\frac{\partial f}{\partial x_0}$ contributes only a small, sometimes negative, impact on the model's output. Therefore, completely removing the bias in model interpretation methods can lead to misjudging the contribution of features. Most current popular research methods share a common flaw: they exclude bias in interpretation results.

In this context, Srinivas and Fleure [14] developed the Full-Gradient method, which focuses on leveraging the impact of bias to enhance the expressiveness of traditional gradient methods by incorporating saliency maps of internal layers within the model. This method utilizes saliency maps throughout the process as the sample moves from the input layer to the classification layer. The saliency map at the initial input layer is defined as $M^{input} = abs(input \times gradient)$, and for subsequent layers, it is $M^{bias} = abs(bias \times gradient)$. The overall aggregation is generally conducted using the formula:

$$M^{Full-Gradient} = M^{input} + \sum M^{bias} \tag{1}$$

3 Our Method

This study uses image classification model as the target to propose the proposed method. This group of models is characterized by consecutive convolutional layers and combined with a classifier composed of one or several densely (or sparsely) connected layers. Besides, the activation function that is often used after going through the layers of the model is ReLU. With these characteristics, we can analyze the model's behavior and form a general formula for the model.

For each stage propagating through a layer of the model, we can think of it as a mapping as follows:

$$x_{l+1} = \psi\left(W_l x_l + b_l\right) \tag{2}$$

In which, ψ is the nonlinear function used for each layer, W_l is the weight, x_l is the input value of the layer and b_l is the bias of the corresponding layer l. This formula can be applied on the fully connected layer and also on the convolutional layer since they have been proven to be equivalent to matrix multiplication.

For an image classification model using l total layers including convolutional layers and clustering, we can rewrite the forward propagation process with the following general formula:

$$x_l = \psi\left(W_{l-1}\psi\left(W_{l-2}...\psi\left(W_0 x_0 + b_0\right)... + b_{l-2}\right) + b_{l-1}\right) \tag{3}$$

When the model uses ReLU as a nonlinear function for each layer, we can decompose the formula 3 into the sum of the two components bias and input by setting $k_l = (W_l x_l + b_l) > 0$. In which $k_l^{i,j} = 1$ if $W_l^{i,j} x_l^{i,j} + b_l^{i,j} > 0$ and $k_l^{i,j} = 0$ if $W_l^{i,j} x_l^{i,j} + b_l^{i,j} \leq 0$. Then, the mapping of a layer on the network using the ReLU activation function becomes

$$\begin{aligned} x_{l+1} &= k_l \odot \left(W_l x_l + b_l\right) \\ &= \left(k_l \odot W_l x_l\right) + \left(k_l \odot b_l\right) \\ &= \left(k_l \odot W_l k_{l-1} \odot W_{l-1} x_{l-1}\right) + \left(k_l \odot W_l k_{l-1} \odot b_{l-1}\right) + \left(k_l \odot b_l\right) \end{aligned} \tag{4}$$

With $\odot$ being the symbol for Hadamard multiplication of 2 matrices, when continuing to implement the formula 3 according to the formula 4 we will get the general equation for The model only uses ReLU as the activation function at the layers

$$x_l = \left(\prod_{i=0}^{l-1} k_i \odot W_i\right) x_0 + \sum_{i=0}^{l-1}\left(\prod_{j=i+1}^{l-1} k_j \odot W_j\right) k_i \odot b_i \tag{5}$$

In which $\prod_i^{l-1} k_i \odot W_i$ is the derivative value $\frac{\partial x_l}{\partial x_i}$ of the model result according to input x_i at layer i of the model, so the formula 4 can be shortened to the following

$$x_l = \frac{\partial x_l}{\partial x_0} x_0 + \sum_{i=0}^{l-1} \left(\frac{\partial x_l}{\partial x_{i+1}} \right) k_i \odot b_i \tag{6}$$

The formula 6 allows separating the components that appear, helping us consider some of the following factors that may occur when building an interpretation method for the convolutional model

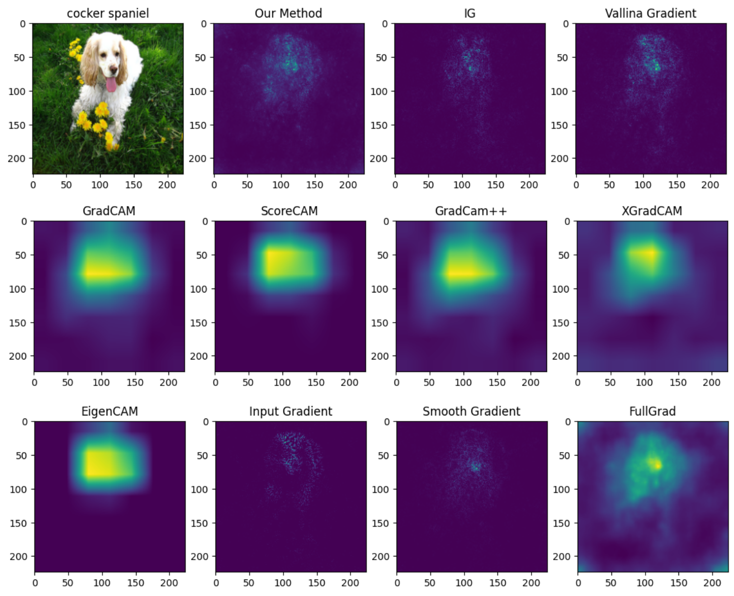

Fig. 1. Visual interpretation of the proposed method compared to previous methods on selected sample images.

Firstly, the influence of bias values appearing in a nonlinear model depends on the previous input value, which, unlike the linear model, is always fixed. Next, using only gradients on the input value of the model input is not complete, or it can be said that they will miss truly influential features that are not detected at the first receiving layer. Finally, methods used on internal input layers within the model tend to be better by reducing the influence of bias on previous layers. Conversely, we need to consider the problem that these methods may arise from the fact that they represent the influence of x_i internally rather than that of x_0. Therefore, the reverse influence mapping x_0 may be flawed by ignoring bias-related issues.

To explain the above problems, we consider a binary nonlinear model with the formula built according to the formula 5.

Bias is not fixed: When we formalize the linear model, we have

$$x_l = \left(\prod_{i=0}^{l-1} W_i\right) x_0 + \sum_{i=0}^{l-1} \left(\prod_{j=i+1}^{l-1} W_j\right) b_i \tag{7}$$

In which W_i is the weight of the model at layer i, because there is no impact of k_l in the formula, the value $\sum_{i=0}^{l-1}\left(\prod_{j=i+1}^{l-1} W_j\right) b_i$ is always a constant for all samples x_0, thus eliminating consideration of bias effects on linear models can be ignored. On the contrary, the formula 6 shows that the impact of bias b_i depends on k_i, which k_i also depends on x_i while x_i also depends on the previous x_{i-1}. Only $k_i \odot b_i$ is subject to variation according to the x_0 pattern, so the impact of bias in the nonlinear model will also vary with each different input value.

A layer itself is incomplete: Methods that use gradient computation, especially the input x gradient method, assume that the model's result is composed entirely of the gradient component on the input sample x_0, so the method is only good when $x_l \approx \frac{\partial x_l}{\partial x_0} x_0$. In other words, they are only good when the bias's contribution is insignificant. And as Wang's research [17] also showed, the value $\frac{\partial x_l}{\partial x_0} x_0$ only plays a small part on x_l. So we cannot express the entire model by just defining it in one layer.

Better representation appears at the inner layers: The formula 6 is intended to perform full network modeling on input x_0, but can still be customized so that they are more suitable to use in other methods, including the popularity of methods that put computation into internal layers of the model, where we can also explain how these methods' representations are somewhat more efficient.

With the method that uses a model with layer $k \to l$ to perform the interpretation process:

$$\begin{aligned} x_{k\to l} &= \frac{\partial x_l}{\partial x_k} x_k + \sum_{i=k}^{l-1} \left(\frac{\partial x_l}{\partial x_{i+1}}\right) k_i \odot b_i \\ &= x_l + \frac{\partial x_l}{\partial x_k} x_k - \frac{\partial x_l}{\partial x_0} x_0 - \sum_{i=0}^{k-1} \left(\frac{\partial x_l}{\partial x_{i+1}}\right) k_i \odot b_i \end{aligned}$$

Note that we have $x_{k\to l} = x_l$. From above, we can see that the formula partially eliminates the impact of bias on the results. When k has a value closer to l, the number of unnecessary biases in the calculation is reduced, leading to $x_{k\to l} \approx \frac{\partial x_l}{\partial x_k} x_k$ or in other words, the dependence of x_l on x_k will be clearer, so the methods the interpretation used on x_k could easily give a better result than when used on x_0. Still, we should note that we are then interpreting the features of x_k rather than of x_0.

To simplify the model including classifiers, we eliminate the process of using the softmax function to give the probability of classifier occurrence in the model, instead, we use the state of the last layer to consider the contribution of the model. The components on the network make up the value of the class we are interested in, in which the value of the class is higher the more characteristics of the class are detected in the model. The model has been decomposed into the total contribution score of the input and the series of total contribution scores of the bias across the layers. Thus, it can be represented by a series of saliency maps throughout the model's processing, in which

$$S^{input}(x) = Sum\left(ReLU\left(\frac{\partial x_l}{\partial x_0} \odot x_0\right)\right)$$
$$S_i^{bias}(x) = Sum\left(ReLU\left(\frac{\partial x_l}{\partial x_{i+1}} \odot (k_i \odot b_i)\right)\right)$$

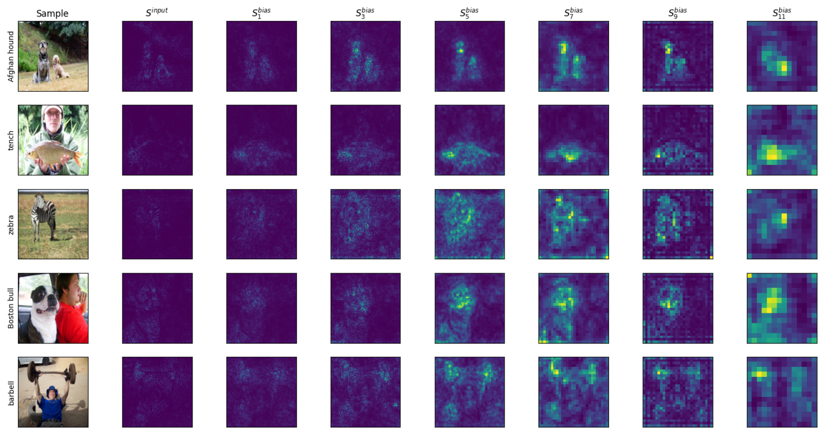

Fig. 2. Visualization of saliency maps throughout the process of the VGG-16 model.

The ReLU function is added so that we can only get the features that positively affect to the result, with the *Sum* function to summarize the total influence on the same location of the image through the extracted feature channels. Figure 2 shows us that if we only consider the receiving layer, the influence distribution is highly dispersed, or even does not appear. When considering bias in the internal layers of the model, features that contribute to the prediction are gradually supplemented by the bias component with information that previous layers may have overlooked.

We can directly use the above saliency maps to consider the components that make up the value of the class, however, to synchronize with other methods and also give a brief explanation, we proceed to combine the saliency maps together into a single map. To do so, we must agree that the model is good enough not to cause image displacement effects, meaning that across all layers, the relative positions of objects in the image are unchanged. For maps whose size is not commensurate with the image size in the receiving layer, we will enlarge them with the bilinear algorithm, then the maps will be normalized with a sum of 1 (linear normalization) and finally received with the weight corresponding to the number of points they contribute to the classification result.

$$\begin{aligned} S(x) &= \phi\left(S^{input}(x)\right) \times score^{input} + \sum \phi\left(S^{bias}(x)\right) \times score^{bias} \\ score^{input} &= ReLU\left(\frac{\partial x_l}{\partial x_0} x_0\right) \\ score_i^{bias} &= ReLU\left(\frac{\partial x_l}{\partial x_{i+1}} (k_i \odot b_i)\right) \end{aligned} \tag{8}$$

Where ϕ are the steps to zoom in and normalize the value of the map. The ReLU function aims to eliminate saliency maps that do not contribute positively to the result. In addition, unlike the Full-Grad [14] method when considering all saliency maps with the same level of influence, adding the level of influence in the above way (score) also aligns the level of influence which shows the positive contribution of the maps when combining them. We can also use them individually instead of combining them together depending on different use cases.

Figure 1 shows us the saliency maps when used on a sample, we can clearly see the difference when comparing the representation between the CAM-based methods and the gradients-based methods. Among them, representational CAM methods have a large segmentation part because of the impact of upsampling the activation layer. Gradient methods are somewhat more focused on the object, but there is still dispersion (or sparsity) due to the surrounding environment. The proposed method provides a somewhat clear visualization of the larger influential features. However, assessing whether the method is intuitively correct or not cannot only be based on objective human judgment.

For each x_l value of a subclass, we can decompose them into 2 separate parts, in which the value of the class part will be composed of positive components, increasing the value of the correct classifier, and negative values, reduce the likelihood of the classifier occurence.

$$s_l = s_l^+ + s_l^- \tag{9}$$

Therefore, the *ReLU* function is applied to each class so that they can eliminate the negative effects of subclassing, keeping only the features that have a positive impact on the model's class recognition. On the contrary, if we use the *Sum* function, the positive and negative effects can cancel each other out, this should not be applied since convolutional networks use multiple channels per layer with each having ability to recognize different patterns in the same area of the object. The *Abs* function will assume that both positive and negative effects should be represented together, which is not necessary and may cause confusion as to which properties make up the value of the analysis class. For the same reason, continuing to use *ReLU* when combining saliency maps also aims to remove layers that do not have a positive effect on the results, avoiding disturbing the overall representation.

4 Experimental results

Most previous methods were implemented using source code provided in the corresponding research papers. However, the proposed method and a few gradient-based methods were implemented by our research team. For example, with the Integrated Gradients method, the baseline used is a 0-valued matrix of the same size as the input sample (the baseline is a black image). In addition, the number of steps for approximation is set by default to 50. The number of samples generated by the Smooth Gradients method is set to 50 by default along with noise taken from a normal distribution with mean at 0 and standard deviation at 0.1. Methods related to upsampling (increasing sample size) are implemented with bilinear algorithm. Finally, with methods using CAM, the target layer defaults to the final feature layer, which is a convolutional layer.

Dataset We use the ImageNet1K dataset [3], which is a subset of the ImageNet dataset, with 1000 different classes. The main object used is the evaluation data set to ensure that pre-trained models have not been learned through the data used for this evaluation. The evaluation data set includes 50,000 images, each class has 50 sample images. The reason our team chose to experiment with this dataset is because ImageNet1K is specifically designed to evaluate the performance of deep learning models across a wide range of object categories. The dataset contains accurately labeled and diverse images, encompassing various real-world scenarios, ensuring the generalizability and reliability of the evaluation results. To enhance objectivity, expedite the experimental process, and facilitate performance comparisons between previous methods and our proposed method, we randomly selected 2,000 sample images from the total of 50,000 images along with their corresponding labels.

Baseline There are 2 groups of methods used as a basis for experimentation. The first is a group of methods using CAM including GradCAM [10], GradCAM++ [2], ScoreCAM [16], EigenCAM [8] and XGradCAM [4]. The remaining group are methods that use gradients including FullGrad [14], Integrated Gradients [15], Smooth Gradients [12], Vanilla Gradient (pure gradient method) and a variation of it using the input as a mask, called Input$\times$Gradient. In the experiments below, the team will flexibly compare the results of the proposed method with all previous methods or with each group of previous methods.

Strategy for generating saliency maps. The input images and the model used are passed to the GPU for the main computational task. This will help experiments to be measured with the same resources and reduce experiment time. Next, the calculation steps related to the saliency map will be processed in batch sizes up to 32.

Evaluation strategy for interpretation methods The baseline used is the image after being blurred according to the Gaussian normal distribution of

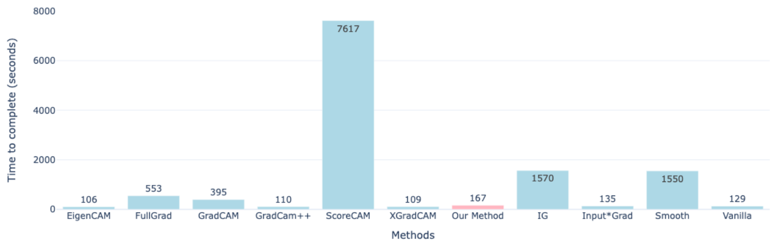

Fig. 3. Comparison chart of generating saliency maps for 2000 images between Our Method and previous methods.

the input sample, including 224 steps of adding or removing features processed in a maximum batch size of 64.

In the context of explanation generation, the speed of producing saliency maps is very important because they will contribute to the interpretation parallel with the model's conclusions. In this aspect, the proposed method shows superiority over ScoreCAM, IG, and Smooth Gradient methods.

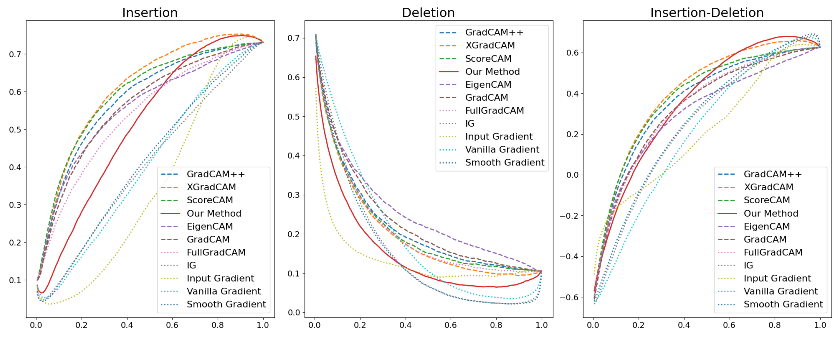

Fig. 4. Comparison chart between previous methods and our proposed method based on Insertion, Deletion, and Insertion-Deletion measures on 2000 images.

Regarding the three metrics Insertion, Deletion, and Insertion-Deletion, the proposed method outperforms the gradient-based algorithm overall. If compared specifically with the baseline method FullGrad, it is almost equally good overall but superior on the deletion measure.

Deletion metric is more reliable In real scenarios, the Insertion metric is not as reliable as the Deletion metric. The reason is that with the Insertion metric, we add pixels one by one to a baseline image. This means that regardless of the added pixels, the baseline part must be categorized as non-feature input. However, CNN models do not have a concept of a baseline input, and there

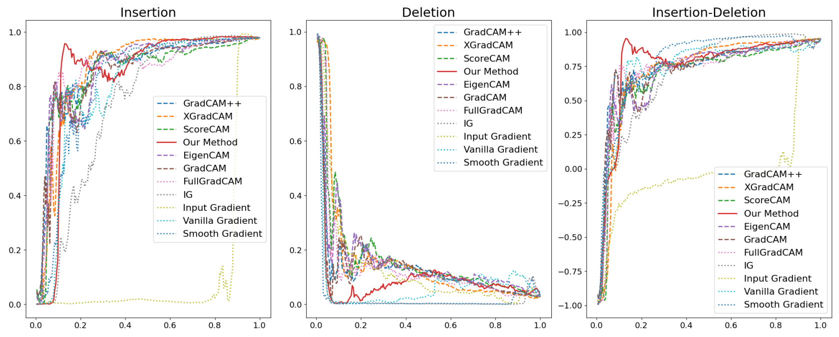

Fig. 5. Comparison chart between previous methods and our proposed method based on Insertion, Deletion, and Insertion-Deletion measures on the sample image.

might be some hidden patterns in the "baseline parts" that we are unaware of and that the models might accidentally learn from. On the other hand, the Deletion metric starts with the full image and removes important patterns one at a time. This directly alters the features that the model has learned, so the greater the drop in confidence, the more it indicates that important features are recognized better.

Regarding the two measures % Increase in Confidence and Average Drop % showed in table 4, the proposed method is at an acceptable level. When compared to CAM methods, the group's method is worse in terms of correlation between reliability increase and saliency map but is better than methods using gradients. However, evaluating the practicality of these two measures is not transparent because of the problem of losing features at the location proposed by the evaluation method.

5 Evaluating Visualizations

Visually, there are some cases where the proposed method performs better than all the other methods, specifically the main object is identified in a very localized way and is not smeared to noisy areas. If we compare visually the explanation in figure 1, it seems that the proposed method localizes image features better than all the other methods. Specifically, with methods using CAM, the image's characteristic area is too wide and not really localized much on the object. Methods that use gradients highlight the main object but do not really appreciate those pixels, so the pixels are a little blurry and the saliency map seems sparse. FullGrad delineates areas on objects more clearly than CAM methods but has a bit more noise, with some areas spreading out to the edges of the image. With the proposed method, the object is localized moderately, and the contribution of the features is highly appreciated, so the pixels are clearer and especially there is no noise.

Table 1. Comparison table between Our Method and previous methods with % Increase in Confidence, Average Drop % metrics

Method	**% Increase in Confidence**	**Average Drop %**
IG	0.10	48.3697
Input×Grad	0.00	51.9696
Vanilla Gradient	0.15	47.2729
Smooth Gradient	0.20	46.4051
FullGrad	20.35	28.6634
GradCAM	24.00	25.8560
GradCAM++	28.10	22.4184
EigenCAM	9.65	26.8552
ScoreCAM	35.25	19.1906
XGradCAM	33.40	15.5054
Our Method	3.05	30.6677

Sanity check For an interpretability method applied to a model, verifying the reasonableness of the visualizations is essential to avoid instances where the method might be solving a different implicit problem. This issue is commonly seen in methods like Guided Backpropagation [13], [5], which can inadvertently turn into edge detection tasks. An acceptable method should be sensitive to the model, meaning that any changes to the model should reflect in the interpretation results. Adebayo [1] proposed a simple method to verify the reasonableness of an interpretability approach by gradually altering the model's weights from the classifier layer to the input layer. If the method is truly model-sensitive, it will exhibit noticeable changes as the model undergoes alterations.

Figure 6 illustrates the results of the method based on the proposed verification approach. It can be observed that as soon as the weights of the linear layers are fine-tuned, the prominent bright regions of the map shift, indicating that the primary focus area appears to move to a different part of the recognized entity. As we move higher up, the areas of influence exhibit greater volatility, most notably during the weight fine-tuning stages of the convolutional layers. In these stages, the map shows no significant features influencing the results, implying that no features extracted by the random convolutional layers positively impact the outcome. With these results, we can conclude that the model is sensitive to changes in its structure.

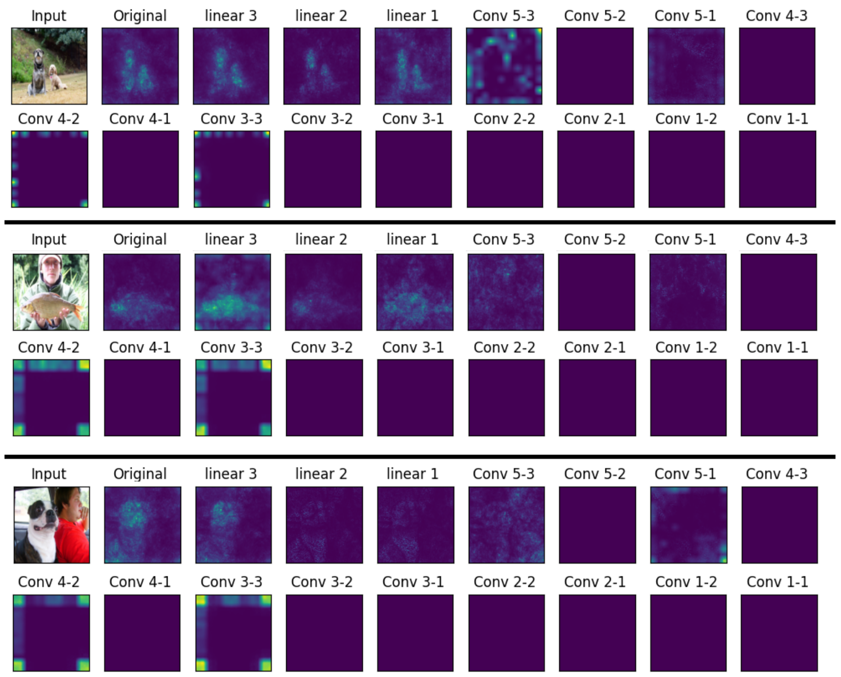

Fig. 6. Sanity check evaluation of our method's visualizations on several samples, resetting the weights of the VGG-16 model incrementally, from the linear layers to the entire model.

6 Discussion

The proposed method offers an approach of incorporating bias into the interpretation contribution instead of relying solely on gradients or CAM. This makes the generated explanation truly faithful to the model's behavior. Visually, the method clearly explains the main focus areas of the model, as well as the related issues of characteristics that can cause errors in prediction and at the same time shows, are correlated in terms of their contribution to forming the model's conclusions.

This method does not primarily aim to enhance interpretability compared to previous methods. Instead, its goal is to provide a comprehensive understanding of the concept related to the contribution of features to model interpretation. Conceptually, the proposed method strengthens the definition of the contribution of model components to draw conclusions, encompassing gradients, bias distributions within each layer, and reliability calculations on saliency maps. While the baseline for the proposed method is FullGrad, the key difference lies in how our approach modifies the computation of saliency maps and assigns weights to each saliency map during their aggregation. This enhancement highlights the truly important pixel regions while partially eliminating noise components.

Limitations The proposed method can only be implemented on networks using nonlinear functions that can be decomposed, such as the ReLU function presented in this work, as well as functions like HardTanh or HardSigmoid. Additionally, using the results of the classifier before applying the softmax function introduces some disadvantages, as it considers the impacts on individual classes separately rather than their interactions, unlike other methods.

7 Conclusion

In this study, we propose a method to create explanations for images by combining two components including gradient and bias. This approach allows the method to interpret the true behavior of the model and achieve higher reliability. The effectiveness of the proposed method has been demonstrated through the ImageNet1K dataset in performance experiments, visual evaluation, and execution speed analysis. Additionally, our method enhances the reliability of saliency maps for interpretation.

Overall, the proposed directions aim to improve the accuracy of understanding the patterns of input samples on which the model is trained to make predictions. This approach also ensures that the model's behavior does not solely depend on any single factor. We will continue to address the aforementioned limitations and weaknesses, while also adapting to and interpreting a wider variety of image classification models beside convolutional models. We aim to further refine the interpretability aspect, making it smoother and capable of highlighting patterns in misclassified samples. This is crucial because, for incorrect predictions, it is important for humans to understand which features the model learned that led to such conclusions.

References

1. Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I.J., Hardt, M., Kim, B.: Sanity checks for saliency maps. In: Neural Information Processing Systems (2018), https://api.semanticscholar.org/CorpusID:52938797 12
2. Chattopadhay, A., Sarkar, A., Howlader, P., Balasubramanian, V.N.: Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 839–847 (2018). https://doi.org/10.1109/WACV.2018.00097 3, 9
3. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. pp. 248–255 (2009). https://doi.org/10.1109/CVPR.2009.5206848 9
4. Fu, R., Hu, Q., Dong, X., Guo, Y., Gao, Y., Li, B.: Axiom-based grad-cam: Towards accurate visualization and explanation of cnns. ArXiv **abs/2008.02312** (2020), https://api.semanticscholar.org/CorpusID:221006223 3, 9
5. Lapuschkin, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLoS ONE **10**, e0130140 (07 2015). https://doi.org/10.1371/journal.pone.0130140 1, 3, 12

6. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. Advances in neural information processing systems **30** (2017) 1
7. Montavon, G., Lapuschkin, S., Binder, A., Samek, W., Müller, K.R.: Explaining nonlinear classification decisions with deep taylor decomposition. Pattern Recognition **65**, 211–222 (2017). https://doi.org/https://doi.org/10.1016/j.patcog.2016.11.008, https://www.sciencedirect.com/science/article/pii/S0031320316303582 1
8. Muhammad, M., Yeasin, M.: Eigen-cam: Class activation map using principal components. pp. 1–7 (07 2020). https://doi.org/10.1109/IJCNN48605.2020.9206626 3, 9
9. 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) 1
10. 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: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 618–626 (2017). https://doi.org/10.1109/ICCV.2017.74 2, 9
11. Shrikumar, A., Greenside, P., Kundaje, A.: Learning important features through propagating activation differences. In: International conference on machine learning. pp. 3145–3153. PMlR (2017) 3
12. Smilkov, D., Thorat, N., Kim, B., Viégas, F., Wattenberg, M.: Smoothgrad: removing noise by adding noise (06 2017) 3, 9
13. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.A.: Striving for simplicity: The all convolutional net. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings (2015), http://arxiv.org/abs/1412.6806 3, 12
14. Srinivas, S., Fleuret, F.: Full-gradient representation for neural network visualization. Advances in neural information processing systems **32** (2019) 3, 8, 9
15. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. In: International conference on machine learning. pp. 3319–3328. PMLR (2017) 9
16. Wang, H., Wang, Z., Du, M., Yang, F., Zhang, Z., Ding, S., Mardziel, P., Hu, X.: Score-cam: Score-weighted visual explanations for convolutional neural networks. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 111–119. IEEE Computer Society, Los Alamitos, CA, USA (6 2020). https://doi.org/10.1109/CVPRW50498.2020.00020, https://doi.ieeecomputersociety.org/10.1109/CVPRW50498.2020.00020 3, 9
17. Wang, S., Zhou, T., Bilmes, J.: Bias also matters: Bias attribution for deep neural network explanation. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 6659–6667. PMLR (6 2019), https://proceedings.mlr.press/v97/wang19p.html 3, 6
18. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2921–2929. IEEE Computer Society, Los Alamitos, CA, USA (6 2016). https://doi.org/10.1109/CVPR.2016.319, https://doi.ieeecomputersociety.org/10.1109/CVPR.2016.319 2