Visual Explanations via Iterated Integrated Attributions

Oren Barkan*1 Yehonatan Elisha*1 Yuval Asher2
Amit Eshel2 Noam Koenigstein2
1The Open University 2Tel Aviv University
https://github.com/iia-iccv23/iia

Abstract

We introduce Iterated Integrated Attributions (IIA) - a generic method for explaining the predictions of vision models. IIA employs iterative integration across the input image, the internal representations generated by the model, and their gradients, yielding precise and focused explanation maps. We demonstrate the effectiveness of IIA through comprehensive evaluations across various tasks, datasets, and network architectures. Our results showcase that IIA produces accurate explanation maps, outperforming other state-of-the-art explanation techniques.

1. Introduction

The emergence of deep learning has ushered in significant breakthroughs within the realm of artificial intelligence, particularly in computer vision. Advanced deep Convolutional Neural Networks (CNNs) architectures [50, 30, 32, 41], and recent Vision Transformer (ViT) models [20, 28] have demonstrated state-of-the-art performance in image classification [37, 50], object detection [29, 17, 10], and semantic segmentation [17, 4] tasks. Yet, many deep learning models lack interpretability, making it difficult to explain the reasoning behind their predictions. As a result, Explainable AI (XAI) has become a prominent research area in computer vision, and numerous methods have been proposed for explaining and interpreting the internal workings of different neural network architectures in various application domains [60, 49, 46, 15, 7, 44, 6, 8, 26].

Explanation methods attempt to produce an explanation map in the form of a heatmap (also known as relevance or saliency map) that attributes the prediction to the input by highlighting specific regions in the input image. Early gradient-based methods produced explanation maps based on the gradient of the prediction w.r.t. the input image [49, 50, 52]. Then, Grad-CAM [46] and the follow-up works by [12, 33, 5] proposed to compute the explanation maps based on the internal activation maps (also known as Class Activation Maps (CAM)) and their corresponding gradients. In parallel, path integration methods such as Integrated Gradients (IG) [54] proposed to produce an explanation map by accumulating the gradients of the linear interpolations between the input and reference images. The aforementioned techniques were formulated and evaluated on CNNs. Following the advent of Transformer-based architectures [55], a variety of approaches has also been proposed for interpreting Vision Transformer (ViT) models [15, 56, 14].

This paper presents Iterated Integrated Attributions (IIA) - a universal technique for explaining vision models, applicable to both CNN and ViT architectures. IIA employs iterative integration across the input image, the internal representations generated by the model, and their gradients. Thereby, IIA leverages information from the activation (or attention) maps created by all network layers, including those from the input image. We present comprehensive objective and subjective evaluations that demonstrate the effectiveness of IIA in generating faithful explanations for both CNN and ViT models. Our results show that IIA outperforms current state-of-the-art methods on various explanation and segmentation tests across all datasets, model architectures, and metrics.

2. Related Work

Explaining CNNs Explanation methods for CNNs have been studied extensively. Saliency-based methods [16, 49, 43, 62, 60, 61] and activation-based methods [21] use the feature-maps obtained by forward propagation in order to interpret the output prediction. Perturbation-based methods [23, 24] measures the output’s sensitivity w.r.t. the input using random perturbations applied in the input space. Gradient methods produce explanation maps based on the gradient itself or via a function that combines the activation maps with their gradients [48, 53]. A prominent example is the Grad-CAM (GC) [46] method that uses the pooled gradients and the activation maps to produce explanation maps. GC
attributed much attention from the XAI community with several follow-up works [12, 25, 33, 5]. Another relevant line of work is path integration methods. Integrated Gradients (IG) [54] integrates over the interpolated image gradients. Blur IG (BIG) [59] is concerned with the introduction of information using a baseline and opts to use a path that progressively removes Gaussian blur from the attributed image. Guided IG (GIG) [36] improves upon IG by introducing the idea of an adaptive path method. By calculating the integration along a different path, high gradient areas are avoided which often leads to an overall reduction in irrelevant attributions. Differing from IG, GIG, and BIG, IIA employs iterated integration, enabling interpolation of the complete set of activation (attention) maps across all network layers. Moreover, IIA does not limit the integrand to plain gradients but accommodates any arbitrary function involving the activation (attention) maps and their gradients. Gradient-free methods produce explanation maps via manipulation over the activation maps without relying on gradient information [57, 19]. For instance, LIFT-CAM employs the DeepLIFT [47] technique to estimate the activation maps’ SHAP values [42], which are then combined with the activation maps to produce the explanation map. However, since these methods do not consider gradient information, their ability to effectively guide explanations towards the predicted class is limited.

Explaining ViTs Initial attempts to interpret Transformers utilized the inherent attention scores of ViT models to gain insights into input processing [55, 11]. However, the challenge lay in effectively combining scores from different layers. Simple averaging of attention scores for each token, for instance, often resulted in signal blurring [1, 15]. Abnar and Zuidema introduced the Rollout method, which computes attention scores for input tokens at each layer by introducing raw attention scores within a layer as well as those from preceding layers [1]. Rollout showed improvements over the use of a single attention layer, but its reliance on simplistic aggregation assumptions often led to highlighting irrelevant tokens. LRP [3], proposed to propagate gradients from the output layer to the beginning, considering all the components in the transformer’s layers beyond the attention layers. Chefer et al. [15] presented Transformer Attribution (T-Attr), a class-specific Deep Taylor Decomposition method in which relevance propagation is applied for positive and negative attributions. More recently, the authors introduced Generic Attention Explainability (GAE) [14], a generalization of T-Attr for explaining Bi-Modal transformers. Both T-Attr and GAE are considered state-of-the-art methods for explaining ViT models and have been shown to outperform multiple strong baselines such as LRP, partial LRP [56], ViT-GC [15], and Rollout [1]. IIA distinguishes itself from the aforementioned approaches in three key ways: Firstly, IIA introduces and utilizes the Gradient Rollout (GR) - a variant of Rollout that combines attention matrices with their gradients. Secondly, IIA employs GR as the integrand in its iterative integration process, conducting integration across interpolated attention matrices. Lastly, IIA stands out as a universal method, capable of generating explanations for both CNNs and ViTs.

3. Iterated Integrated Attributions

We start by describing the problem setup. Then, we briefly overview IG [54] and continue to describe IIA in detail.

3.1. Problem Setup

Let \( x \in \mathbb{R}^{q_0 \times p_0 \times q_0} \) be an input image. We define a generic neural network model with \( L \) intermediate layers, each is a function \( h_l \) \((1 \leq l \leq L)\) that outputs \( x^l := h_l(x^{l-1}) \), with \( x^0 := x \). The final layer is a classification head \( f \) that produces the prediction \( f(x^L) \), and the score for the class \( y \) is given by \( f_y(x^L) \). Additionally, we define the application of the neural network to the input \( x \) by

\[
\phi(x) = f(x^L). \tag{1}
\]

For example, if \( \phi \) is a ResNet (ViT) model, each \( h_l \) would be implemented as a residual convolutional (transformer encoder) block. Our goal is to generate an explanation map \( m \in \mathbb{R}^{q_0 \times p_0 \times q_0} \) that quantifies the attribution of each element in \( x \) to the prediction \( \phi(x) \). The attribution can be computed w.r.t. \( \phi_y(x) \) - the score assigned to the class \( y \). Typically, the class of interest \( y \) is either set to the target (ground-truth) class or to the predicted class, which is the class receiving the highest score in \( \phi(x) \).

3.2. IG

In what follows, we quickly overview IG [54], which is a special case of IIA. Given the input \( x \) and a reference \( r \in \mathbb{R}^{q_0 \times p_0 \times q_0} \) (that is designed to represent missing information, hence usually set to the zero image), we define a linear interpolant by

\[
v = r + a(x - r), \tag{2}\]

with \( a \in [0, 1] \). IG produces an explanation map by integrating gradients along the linear path between \( r \) and \( x \) as follows:

\[
m_{IG} = \int_0^1 \frac{\partial \phi_y(v)}{\partial v} \circ \frac{\partial v}{\partial a} da = (x - r) \circ \int_0^1 \frac{\partial \phi_y(v)}{\partial v} da, \tag{3}\]

where \( \circ \) stands for the Hadamard (elementwise) product. In practice, the integral in Eq. 3 is numerically approximated as follows:

\[
m_{IG} \approx \frac{x - r}{n} \circ \sum_{k=1}^n \frac{\partial \phi_y(v)}{\partial v}, \tag{4}\]

2074
by setting \( a = \frac{k}{n} \) in Eq. 2. The approximation in Eq. 4 simply sums the gradients of \( n \) interpolants on the linear path from \( r \) to \( x \). Finally, since \( m_{ij} \) is in \( \mathbb{R}^{c_0 \times p_0 \times q_0} \) (typically, \( c_0 = 3 \) since \( x \) is a RGB image), mean reduction along the channel axis is performed to obtain a 2D explanation map.

### 3.3. IIA - A Generic Formulation

IIA diverges from IG in several aspects: First, IIA does not confine gradient computation to the input \( x \). In fact, recent studies have suggested that gradients derived from internal activation maps can yield improved explanation maps \([46, 12, 25]\). Secondly, IIA employs an iterated integral across multiple intermediate representations (such as activation or attention maps) generated during the network’s forward pass. This enables the iterative accumulation of gradients w.r.t. the representations of interest. Lastly, unlike IG, IIA does not restrict the integrand to plain gradients, but encompasses a function of the entire set of representations produced by the network and their gradients. In this section, we assume a generic neural network model. In Sec. 3.4, we describe the utilization of IIA for CNN and ViT models.

As outlined above, IIA utilizes linear interpolations on the intermediate representations generated during the forward propagation of the input through the layers of the model. In order to incorporate interpolation, we modify the computation in the \( l \)-th layer to accommodate an interpolation to the intermediate representation of interest (produced as part of the computational pipeline of \( h^l \)). To facilitate the formulation of the IIA approach, we introduce a set of notations: First, the input to the \( l \)-th layer undergoes processing by a function \( u^l \) to obtain the intermediate representation of interest, denoted as \( u^l \). Subsequently, an interpolation step is (optionally) performed to derive the interpolant \( v^l \) (interpolated version of \( u^l \)). Finally, the interpolant \( v^l \) is processed by a function \( v^l \) that completes the original computational pipeline, yielding the input to the subsequent layer in the model. This entire process can be expressed mathematically using the following equations:

\[
h^l = v^l(v^l),
\]

with

\[
v^l = r^l + (a_l)^{b_l}(u^l - r^l),
\]

and

\[
u^l = u^l(h^{l-1}).
\]

The rationale behind Eqs. 5-7 is as follows: \( u^l \) is a function that computes the intermediate representation of interest \( u^l \) (the representation that is to be interpolated) based on the input to the \( l \)-th layer \( h^{l-1} \). \( u^l \) is further subtracted by a corresponding reference \( 1 \) representation \( r^l = \min(u^l) \), which is the minimum value in each channel of \( u^l \) that is subsequently broadcast to a tensor with the same dimensions as \( u^l \). Additionally, in Eq. 6, \( b_l \) is an indicator parameter that determines whether the interpolation is effectively applied to \( u^l \) during the propagation via the \( l \)-th layer in the model (\( b_l = 1 \)) or not (\( b_l = 0 \)), and \( a_l \in [0, 1] \) controls the interpolation step, hence playing a similar role as \( a \) from Eq. 3, resulting in the interpolant \( v^l \). Finally, \( v^l \) is a function that receives the (interpolated) intermediate representation \( v^l \) and completes the required computation for producing the expected output from the \( l \)-th layer. Hence, the dimensions of \( h^l \) must match those of \( h^l(x^{l-1}) \). Moreover, if \( b_l = 0 \) for all \( j \leq l, u^l = h^l \) and \( v^l \) is the identity mapping, then \( h^l \) and \( h^l(x^{l-1}) \) are identical. Note that the implementation of \( u^l \), \( v^l \), and the choice of representations to be interpolated all vary based on the model’s architecture (as will be detailed in Sec. 3.4).

We further define \( b = [b_0, ..., b_L] \in \{0, 1\}^{L+1}, h^{-1} = x \), and set \( u^0 \) and \( v^0 \) to the identity mapping. Therefore, we have

\[
u^0 = x \quad \text{and} \quad h^{0} = v^{0}.
\]

Finally, the IIA explanation map is defined as follows:

\[
m_{b}^{l} = \int_{0}^{1} \int_{0}^{1} \cdots (u^{l} - r^{l}) \circ \int_{0}^{1} q^{l} da_{l} \cdots da_{0}, \tag{9}
\]

where the integrand \( q^{l} \) is a function of the first \( l \) intermediate representations produced by the model (including the input representation) and their gradients.

It is worth exploring the versatility of Eq. 9: \( q^{l} \) determines the integrand that is a function of the participating representations and their gradients from the first \( l \) layers in the model. \( b \) determines which of the representations produced by the first \( l \) layers are effectively interpolated: if \( b_{j} = 1 \ (0 \leq j \leq l) \), then the integration is effectively applied w.r.t. the variable \( a_{j} \), otherwise \( b_{j} = 0 \) and both Eqs. 6 and 9 become agnostic to \( a_{j} \). For example, one can observe that by setting \( b_{0} = [1, 0, ..., 0], l = 0, w^{l} = h^{l} \) (for \( l > 0 \)), \( q^{l} = \frac{\partial f_{a}(h^{l})}{\partial v^{l}} \), and \( v^{l} \) to the identity mapping, Eq. 9 (IIA) degenerates to Eq. 3 (IG) as follows:

\[
m_{b_{0}}^{0} = (u^{0} - r^{0}) \circ \int_{0}^{1} \frac{\partial f_{a}(h^{L})}{\partial v^{0}} da_{0} = (x - r^{0}) \circ \int_{0}^{1} \frac{\partial f_{a}(v^{0})}{\partial v^{0}} da_{0}.
\]

The first equality follows from Eq. 9, and the second is due to Eqs. 1 and 8. Finally, by dropping the zero index, we receive Eq. 3.

IIA (Eq. 9) provides the freedom to run over multiple interpolated representations (including the input) in an iterative manner. Once \( l \) is set, the integrand \( q^{l} \) changes based on the interpolated representations \( h^{l} (j \leq l) \) in the preceding
layers that participate in the interpolation process, where participation is determined by the indicator vector $b$. For example, if we set $l = L$ and $b = [1, 1, \ldots, 1]$, $m^L_b$ will be the outcome of a $L$ iterated integrals over $q^L$. Thus, in computing $m^L_b$, all the intermediate representations within the network (including the input) are iteratively interpolated.

In practice, $m^L_b$ is numerically approximated using:

$$m^L_b \approx \frac{1}{n} \sum_{k=1}^{n} \frac{1}{k} \sum_{k_1=1}^{n} \ldots \frac{1}{n} (u^l - r^l) \circ \sum_{k_1=1}^{n} q^l$$

$$= \frac{1}{n^L} \sum_{k=1}^{n} \sum_{k_1=1}^{n} \ldots (u^l - r^l) \circ \sum_{k_1=1}^{n} q^l,$$

Equation (10)

where $\beta = \sum_{i=0}^{l} b_i$, and $a_j = \frac{b_j}{n}$ (Eq. 6). Again, Eq. 10 degenerates to Eq. 4 for $b_0 = [1, 0, \ldots, 0]$ and $l = 0$. Note that if $q^l$ is not a 2D tensor, a subsequent mean reduction step is required to obtain a 2D explanation map (followed by a resize operation to align with the spatial dimensions of the input $x$, if needed).

3.4. IIA Implementation

CNN Models In CNNs, $\phi$ follows a CNN architecture (e.g., ResNet [31]). In this case, all $h^l$ are residual convolutional blocks producing 3D tensors, i.e., activation maps. In our implementation, we choose to apply the interpolation on the activation maps, hence we set $u^l$ to the identity mapping, $u^l = h^l$, and the $\min$ reduction operation in the computation of $r^l$ is applied channel-wise (followed by broadcasting). Additionally, we set the integrand $q^l = v^l \cdot \frac{\partial f(h^L)}{\partial v}$. The motivation for this choice is as follows: $v^l$ is the (interpolated) activation map that highlights regions where filters are activated, facilitating pattern detection. Its gradient quantifies the attribution level of the particular class of interest to each element in the activation map. Thus, we anticipate that areas where both the gradient and activation exhibit substantial magnitude with a consistent sign will yield effective explanations. This characteristic is achieved through the Hadamard product between $v^l$ and its gradient. Finally, we apply a mean reduction to the channel axis, followed by a resize operation to obtain a 2D explanation map.

ViT Models In the case of ViT [20], the input $x$ is a 2D tensor corresponding to a sequence of tokens (vectors), where the first token is the [CLS] token, and the rest represent patches from the input image. In our implementation, we opted to interpolate the attention matrices. To this end, we set $u^l$ to the attention function which involves the softmax operation on the scaled dot-product between the query and key representations across multiple attention heads. Assuming there are $p$ attention heads, for each head, we perform interpolation on the attention matrix. In this process, the reference $r^l$ is assigned as the zero tensor since all entries in the attention matrices are positive due to the softmax operation. Accordingly, $r^l$ continues the self-attention computation by multiplying the interpolated attention matrices with the value representations for each head. This is followed by the necessary computational steps that generate a new set of token representations for the subsequent transformer encoder layer [20]. Finally, we propose setting the integrand $q^l$ to the Gradient Rollout (GR) - a variant of the Attention Rollout (AR) method [1]. Similarly to AR, GR amalgamates information from the [CLS] attention across all attention heads in the model. However, with a notable distinction, each (interpolated) attention matrix is substituted by the Hadamard product of the attention matrix and its corresponding gradient. The exact implementation of GR is detailed in our git repository. Given that the output of GR is already in the form of a 2D tensor, only a subsequent resize operation is necessary to achieve an explanation map that corresponds to the spatial dimensions of the input image. Finally, it is noteworthy that our experimentation indicates that replacing the matrix product operation with the matrix sum (as part of the GR computation) leads to comparable performance.

Due to the large combinatorial space ($2^L$ possible combinations for $b$), and the fact we evaluate on large models, in this work, we consider double and triple integration in our complete experiments.

For double integration (IIA2), we set $l = L$, and $b = [1, 0, 0, \ldots, 0, 0, 1]$, in Eq. 10, i.e., $b_0 = 1$ and $b_L = 1$, and the rest $b_l = 0 (1 < l < L)$. This means IIA effectively interpolates over the input image and the activation (attention) maps from the last layer in the CNN (ViT) model. Interpolating on the input image, enables us to examine various interpolations of the image and study the significance of pixel features along the integration path. Moreover, integrating on the last layer allows us to explore the importance of the aggregated information from the different layers of the network, as it combines all the network’s features.

For triple integration (IIA3), we further interpolate on the penultimate layer $L - 1$, i.e., $b_0 = 1$, $b_{L-1} = 1$, $b_L = 1$, and the rest $b_l = 0 (1 < l < L)$. This is motivated by the fact that the penultimate layer captures more comprehensive objects and features, as it is closer to the classification head. By including a broader aggregation of features, it assists in predicting specific classes. In contrast, earlier layers primarily focus on detecting low-level features such as edges.

Finally, for both IIA2 and IIA3, we set $n = 10$ and $l = L$ in Eq. 10, i.e., 10 interpolation steps for each selected layer, and the integrand is computed w.r.t. the last layer.

3.5. Computational Complexity

The computational complexity of IIA is determined by the order of the iterated integral being computed. We uti-
lized the approximation from Eq. 10, which is based on nested sums (each comprising \( n \) terms). Each term necessitates the application of \( q_i \), whose computational complexity varies based on the specific implementation. For instance, in Sec. 3.4, \( q_i \) combines both activation (attention) maps and their gradients, leading to computations involving both forward and backward passes. Therefore, if the computational complexity of \( q_i \) is \( \mathcal{O}(Q) \), the overall computational complexity of Eq. 10 is \( \mathcal{O}(n^3Q) \). Yet, the complexity induced by \( n^3 \) can be significantly reduced through the utilization of batch processing via GPUs. For example, in IIA2 (iterated integration on the input and the last layer), performing \( n \) interpolations on the input in a batch is straightforward. Next, we can extend this process to internal layers: creating batches for all interpolations of each activation map, concatenating these batches into a single internal layers: creating batches for all interpolations of each layer), performing \( n \) interpolations on the input in a single forward-backward pass. Therefore, if the number of layers where interpolation is effectively applied (e.g., in IIA2, \( M = 2 \)). Note that the first and second terms in \( R(IIA_M) \) are the costs of the forward and backward passes, respectively. Assuming a GPU with \( B \geq n^M \), it follows that \( \frac{n^m}{B^m} = \mathcal{O}(1) \) for all \( 1 \leq m \leq M \), resulting in the cost of \( IIA_M \) being bounded by a single forward-backward pass. For example, for IIA2 and IIA3 with \( n = 10 \), having \( B = 100 \) and \( B = 1000 \), respectively, is adequate to achieve \( \frac{n^m}{B^m} = \mathcal{O}(1) \), which should be manageable with a high performance GPU. In these scenarios, the runtimes of GC, IG, and IIA are comparable. Theoretically, if \( B \geq n^M \), IIA can become faster than IG, since in IG the gradients are backpropagated through the entire network back to the input, while in IIA2 gradients are backpropagated to the layer \( i_M \) (usually one of the penultimate layers). Lastly, distributing IIA computations across multiple machines can yield further speed-up.

4. Experimental Setup and Results

Our evaluation include five models: ViT-Base (ViT-B), ViT-Small (ViT-S) [20], ResNet101 (RN) [30], DenseNet201 (DN) [32], and ConvNext-Base (CN) [41]. Preprocessing details and links to all models are provided in our GitHub repository.

Evaluation Tasks and Metrics We present an extensive evaluation of both explanation and segmentation tasks. It is worth noting that having superior segmentation accuracy does not necessarily equate to having superior explanatory proficiency. Nevertheless, we conduct segmentation tests to ensure comprehensive comparison with previous works [15, 14, 33, 58]. The explanation metrics include Area Under the Curves (AUCs) of Positive (POS) and Negative (NEG) perturbations tests [15], AUC of the Insertion (INS) and Deletion (DEL) tests [45], AUC of the Softmax Information Curve (SIC) and Accuracy Information Curve (AIC) [35], Average Drop Percentage (ADP), and Percentage Increase in Confidence (PIC) [12]. For POS, DEL, and ADP the lower the better, while for NEG, INS, SIC, AIC, and PIC the higher the better. The segmentation metrics include Pixel Accuracy (PA), mean-intersection-over-union (mIoU), mean-average-precision (mAP), and the mean-F1 score (mF1) [15]. A detailed description of the metrics is provided in Appendix A. Finally, in Appendix C, we provide extensive evaluation on sanity tests [2] that further validate IIA as a machinery for generating faithful explanation maps.

Datasets Explanation maps are produced for the ImageNet [18] ILSVRC 2012 (IN) validation set, consisting of 50K images from 1000 classes. We follow the same setup from [15], where for each image, an explanation map is produced twice: (1) w.r.t. the ground-truth class (Target) and (2) w.r.t. the class predicted by the model (Predicted), i.e., the class that received the highest score. Accordingly, results are reported for both the predicted and target classes. Segmentation tests are conducted on three datasets: (1) ImageNet-Segmentation [27] (IN-Seg): This is a subset of ImageNet validation set consisting of 4,276 images from 445 classes for which annotated segmentations are available. (2) Microsoft Common Objects in COntext 2017 [39] (COCO): This is a validation set that contains 5,000 annotated segmentation images from 80 different classes. Some images consist of multi-label annotations (multiple annotated objects). In our evaluation, all annotated objects in the image are considered as the ground-truth. (3) PASCAL Visual Object Classes 2012 [22] (VOC): A validation set that contains annotated segmentations for 1,449 images from 20 classes.

Evaluated Methods and Hyperparameter Setting The following explanation methods for CNN models are evaluated as baselines: (1) Grad-CAM (GC) [46]. (2) Grad-CAM++ (GC++) [12]. (3) FullGrad (FG) [53]. (4) Ablation-CAM (AC) [19]. (5) Layer-CAM (LC) [33]. (6) LIFT-CAM (LIFT) [34], a state-of-the-art method that was shown to outperform other strong baselines like ScoreCAM [57]. (7) Integrated Gradients (IG) [54]. (8) Guided IG (GIG) [36]. (9) Blur IG (BIG) [59]. (10) X-Grad-CAM (XGC) [25]. For ViT models, we considered the following two methods: (11) Transformer Attribution (T-Attr) [15], a state-of-the-art method that was shown to outperform a variety of other strong baselines such as LRP [3], partial LRP [56], Raw-Attention [14], GC [14] for transformers, and Rollout. (12)
Figure 1. Explanation maps produced for IIA (IIA3) and three path integral baselines using CN w.r.t. the ‘Kerry blue terrier’ (top) and ‘tailed frog, bell toad, ribbed toad, tailed toad, Ascaphus trui’ (bottom) classes.

Generic Attention Explainability (GAE) [14] - This is another state-of-the-art method that was shown to outperform T-Attr on several metrics. When applied, hyperparameters for all methods were configured according to the recommended configuration by the authors. (13) Our generic IIA method is evaluated on both CNN and ViT models. A detailed description of the baselines is provided in Appendix B.

4.1. Results

Explanation Tests  Tables 1 and 2 present explanation tests for CNN and ViT models, respectively. The results encompass all combinations of datasets, models, explanation methods, explanation metrics, and settings (target and predicted). Notably, IIA consistently outperforms all baselines across all metrics and architectures. Among the IIA variants, IIA3 surpasses IIA2 for both ViTs (Tab. 2) and CNNs (Tab. 1, holding true for the vast majority of model-metric combinations). This trend underscores the advantage of triple integration, which incorporates information from layer $L - 1$. In CNNs, GC and GC++ are the runner up, utilizing both activation and gradients, and outperforming other methods across most metrics. Furthermore, path integration methods (IG, BIG, and GIG) exhibit competitive results on POS and DEL metrics but demonstrate weaker performance on other metrics. This divergence could be attributed to the granular output maps generated by integration-based methods, as depicted in Fig. 1. These methods focus solely on integration within the input space, ignoring activations, and thus might overlook key features. Notably, achieving high performance on POS and poor performance on NEG metrics reinforces this observation. As path integration methods yield sparse maps that may impact performance in certain metrics, we also report results for the SIC and AIC metrics [35], employed in the evaluation of GIG[36] and BIG[59]. However, the inclusion of SIC and AIC metrics does not alter the observed trends in the results. This finding emphasizes that IIA is exceptionally effective in generating high-quality explanation maps.

Segmentation Tests  Tables 3 and 4 present segmentation tests results on CNN and ViT models, respectively. The results are reported for all combinations of datasets, models, explanation methods, and segmentation metrics. For these experiments, we exclusively consider the top 5 performing CNN explanation methods from Tab. 1. Once again, it is evident that IIA is the best performer, yielding the most accurate segmentation results for both CNN and ViT models.

Qualitative Evaluation  Figures 2 and 3 present a qualitative comparison of the explanation maps obtained by the top-performing CNN explanation methods and ViT explanation methods, respectively. These examples are randomly selected from multiple classes within the IN dataset. Arguably, IIA (IIA3) produces the most accurate explanation maps in terms of class discrimination and localization. These results align well with the trends observed in Tabs. 1-4. For example, in Fig. 2, IA distinguishes itself by capturing multiple objects related to the target class, setting it apart from the other methods. We further observe that in the case of class ‘accordion, piano accordion, and squeeze box’, IIA focuses mostly on the correct item, while the gradient-free methods like AC and LIFT focus mostly on different parts of the image, showcasing their class-agnostic behavior. Interestingly, in the second row, LIFT generates a flat explanation map, a phenomenon warranting further investigation in future research. Additional qualitative results for both CNN and ViT models are provided in Appendix E.

4.2. Ablation Study

In this work, we employ IIA with double and triple integrals. In this section, we investigate the contribution and necessity of these choices. To this end, we consider three alternatives: (1) IMG - only the input image is interpolated, i.e., we set $b_0 = 1$ and $b_j = 0$ for all $j > 0$. (2) ACT - only the representation (activation or attention maps) produced by the $L$-th layer in the model is interpolated, i.e., we set $b_L = 1$
<table>
<thead>
<tr>
<th>GC</th>
<th>GC++</th>
<th>LIFT</th>
<th>AC</th>
<th>IG</th>
<th>GIG</th>
<th>BIG</th>
<th>FG</th>
<th>LC</th>
<th>XGC</th>
<th>IIA2</th>
<th>IIA3</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEG Predicted</td>
<td>56.41</td>
<td>55.20</td>
<td>55.39</td>
<td>54.98</td>
<td>45.66</td>
<td>43.97</td>
<td>42.25</td>
<td>54.81</td>
<td>53.52</td>
<td>53.46</td>
<td>56.29</td>
</tr>
<tr>
<td>Target</td>
<td>55.34</td>
<td>55.95</td>
<td>55.23</td>
<td>55.46</td>
<td>42.02</td>
<td>41.93</td>
<td>41.22</td>
<td>54.65</td>
<td>54.19</td>
<td>53.93</td>
<td>56.22</td>
</tr>
<tr>
<td>POS Predicted</td>
<td>17.82</td>
<td>18.01</td>
<td>17.53</td>
<td>19.38</td>
<td>17.24</td>
<td>17.68</td>
<td>17.44</td>
<td>18.06</td>
<td>17.92</td>
<td>21.02</td>
<td>16.62</td>
</tr>
<tr>
<td>Target</td>
<td>17.65</td>
<td>18.12</td>
<td>17.48</td>
<td>19.73</td>
<td>16.93</td>
<td>17.48</td>
<td>17.26</td>
<td>17.89</td>
<td>17.79</td>
<td>20.61</td>
<td>16.69</td>
</tr>
<tr>
<td>INS Predicted</td>
<td>48.14</td>
<td>47.56</td>
<td>45.39</td>
<td>47.85</td>
<td>39.87</td>
<td>37.92</td>
<td>36.04</td>
<td>42.68</td>
<td>46.11</td>
<td>43.26</td>
<td>48.01</td>
</tr>
<tr>
<td>Target</td>
<td>48.22</td>
<td>47.27</td>
<td>44.94</td>
<td>47.75</td>
<td>37.55</td>
<td>34.41</td>
<td>34.68</td>
<td>43.08</td>
<td>45.91</td>
<td>43.13</td>
<td>48.05</td>
</tr>
<tr>
<td>RN Predicted</td>
<td>17.87</td>
<td>16.91</td>
<td>18.03</td>
<td>16.19</td>
<td>37.52</td>
<td>35.28</td>
<td>38.05</td>
<td>21.06</td>
<td>24.34</td>
<td>17.02</td>
<td>12.79</td>
</tr>
<tr>
<td>Target</td>
<td>17.83</td>
<td>15.97</td>
<td>17.36</td>
<td>15.30</td>
<td>36.51</td>
<td>36.00</td>
<td>41.98</td>
<td>20.29</td>
<td>23.78</td>
<td>16.39</td>
<td>12.31</td>
</tr>
<tr>
<td>ADP Predicted</td>
<td>36.69</td>
<td>36.53</td>
<td>35.95</td>
<td>35.52</td>
<td>19.94</td>
<td>18.72</td>
<td>24.53</td>
<td>31.59</td>
<td>35.43</td>
<td>36.18</td>
<td>42.96</td>
</tr>
<tr>
<td>Target</td>
<td>37.84</td>
<td>38.37</td>
<td>37.64</td>
<td>37.31</td>
<td>21.43</td>
<td>15.81</td>
<td>23.94</td>
<td>33.18</td>
<td>35.64</td>
<td>37.54</td>
<td>45.21</td>
</tr>
<tr>
<td>PIC Predicted</td>
<td>76.91</td>
<td>76.44</td>
<td>76.73</td>
<td>73.36</td>
<td>54.67</td>
<td>55.04</td>
<td>56.98</td>
<td>75.35</td>
<td>73.93</td>
<td>72.64</td>
<td>78.52</td>
</tr>
<tr>
<td>Target</td>
<td>76.87</td>
<td>76.62</td>
<td>76.81</td>
<td>75.55</td>
<td>51.54</td>
<td>54.87</td>
<td>55.23</td>
<td>75.39</td>
<td>73.71</td>
<td>72.71</td>
<td>78.13</td>
</tr>
<tr>
<td>SIC Predicted</td>
<td>74.36</td>
<td>71.97</td>
<td>72.76</td>
<td>70.35</td>
<td>51.92</td>
<td>53.38</td>
<td>53.64</td>
<td>71.49</td>
<td>67.57</td>
<td>69.85</td>
<td>75.49</td>
</tr>
<tr>
<td>Target</td>
<td>72.49</td>
<td>71.42</td>
<td>73.45</td>
<td>70.48</td>
<td>52.71</td>
<td>52.54</td>
<td>54.24</td>
<td>71.38</td>
<td>66.18</td>
<td>70.24</td>
<td>75.88</td>
</tr>
</tbody>
</table>

Table 1. Explanation tests results on the IN dataset (CNN models): For POS, DEL and ADP, lower is better. For NEG, INS, PIC, SIC and AIC, higher is better. See Sec. 4 for details.
Table 2. Explanation tests results on the IN dataset (ViT models):

We set the penultimate layer, which has a higher spatial feature map resolution. For POS, DEL, and ADP, lower is better. For NEG, INS, PIC, SIC, and AIC, higher is better. See Sec. 4 for details.

Table 3. Segmentation tests on three datasets (CNN models). For all metrics, higher is better. See Sec. 4 for details.

and $b_j = 0$ for all $j < L$. Note that for both IMG and ACT, we set $l = L$ in Eq. 10, i.e., the integrand is computed w.r.t. the $L$-th layer. (3) **IIA2 (L-1)** - performs double integral, but interpolates on the layer $L - 1$ instead of the last layer $L$ (by setting $b_0 = 1$, $b_{L-1} = 1$, and $b_j = 0$ for all other layers).

Table 5 reports the results for the RN and ViT-B models on the IN dataset under the target settings. For the sake of completeness, we further include the results for IG, IIA2, and IIA3 (Tabs. 1 and 2). We see that IIA2 and IIA3 perform the best. While ACT is inferior to IIA2, it outperforms IMG. This underscores the need to interpolate on the activations. Yet, the contributions from both IMG and ACT are complementary, as can be seen in IIA2 that combines both.

Interestingly, IIA2 (L-1) outperforms IIA2 and IIA3 in terms of POS and DEL metrics, on the RN model. Figure 4 demonstrates this trend visually. This finding suggests that IIA2 (L-1) generates more focused maps as it utilizes the penultimate layer, which has a higher spatial feature map resolution of $14 \times 14$ (compared to $7 \times 7$ in the last convolutional layer in RN), hence is capable of producing more focused explanation maps that lead to better performance on POS and DEL metrics. This is due to the fact that the deletion of the most relevant pixels results in fewer pixels being removed, and the mask is more focused on a subset of pixels compared to IIA2. IIA2 (that operates on the last layer with lower resolution) produces less focused explanation maps that lead to better performance on POS and DEL metrics.
5. Conclusion

We introduced Iterated Integrated Attributions (IIA) - a universal machinery for generating explanations for vision models. IIA employs iterative accumulation of information from interpolated internal network representations and their gradients. Our experiments highlight IIA’s effectiveness in explaining both CNN and ViT models, consistently outperforming state-of-the-art explanation methods across diverse tasks, datasets, models, and metrics.
References


