Self-Guided Integrated Gradient Method for Attribution

Supplementary Material

A1. SIGMA Pseudocode

Algorithm 1 SIGMA: Self-guided Integrated Gradient Method for Attribution

Input:
- $f(\boldsymbol{\sigma})$: prediction landscape for the model
- $\boldsymbol{\sigma}_{k=0}$: input image to be explained
- f: model confidence for the target class
- α: step size
- β: perturbation magnitude
- ϵ: stopping threshold on model confidence

Initialise: $k = 0$, attribution map $\boldsymbol{\phi}^{\text{SIGMA}} = \mathbf{0}$

while $f(\boldsymbol{\sigma}_k) > \epsilon$ **do**

Sample random perturbation pattern $\boldsymbol{\Delta}_k$

Compute stochastic gradient estimate (SPSA):

$$\nabla f(\boldsymbol{\sigma}_k) = \frac{f(\boldsymbol{\sigma}_k + \beta\boldsymbol{\Delta}_k) - f(\boldsymbol{\sigma}_k - \beta\boldsymbol{\Delta}_k)}{2\beta} \cdot \boldsymbol{\Delta}_k$$

Update path in the direction that reduces confidence:

$$\boldsymbol{\sigma}_{k+1} = \boldsymbol{\sigma}_k - \alpha \cdot \nabla f(\boldsymbol{\sigma}_k)$$

Compute model gradients for the updated image:

$$\boldsymbol{g}_{k+1} = \nabla f(\boldsymbol{\sigma}_{k+1})$$

Compute the confidence drop between successive steps:

$$f(\boldsymbol{\sigma}_k) - f(\boldsymbol{\sigma}_{k+1})$$

Accumulate attribution:

$$\boldsymbol{\phi}_{k+1} = \boldsymbol{g}_{k+1} \times \frac{f(\boldsymbol{\sigma}_k) - f(\boldsymbol{\sigma}_{k+1})}{\sum_{i'} g_{k+1}^{(i')}}$$

$\boldsymbol{\phi}^{\text{SIGMA}} = \boldsymbol{\phi}^{\text{SIGMA}} + \boldsymbol{\phi}_{k+1}$

$k = k + 1$

end while

Output: Final attribution map $\boldsymbol{\phi}^{\text{SIGMA}} = \sum_{k=1}^{N} \boldsymbol{\phi}_k$

A2. Axioms

Section 3.3 introduces one of the axiomatic properties of SIGMA, Completeness, with the path construction ensuring this property is met. Satisfying completeness implies that a second axiom, Sensitivity(a) is also satisfied (see [25] for the full proof), which states that if an input and baseline differ in one feature, and the prediction of the network changes, then that feature should receive a non-zero attribution. Below we provide discussion and proofs on the satisfaction of the axioms Sensitivity (b) and Implementation Invariance. Additionally, we demonstrate that Symmetry is satisfied under specific conditions, which are discussed in detail.

Definition 2 (Sensitivity(b))**.** *Also referred to as the Dummy axiom, as described in [45], states that if a feature is not referenced by the model, meaning the model's output remains invariant under changes to that feature, then it should receive zero attribution.*

Remark 1. *SIGMA satisfies Sensitivity(b), since the partial derivative of the model output with respect to a dummy feature is zero at all points along the attribution path. Consequently, the dummy feature receives zero attribution.*

Proof. Let feature j be a dummy feature on the SIGMA path, meaning that for any change in value from $\sigma_k^{(j)}$ to $\sigma_k^{(j')}$, all other values remain the same $\sigma_k^{(-j)}$, we have

$$f(\sigma_k^{(j)}, \sigma_k^{(-j)}) = f(\sigma_k^{(j')}, \sigma_k^{(-j)}).$$

That is, the output of f is invariant to changes in the j-th feature.

According to Eq. 5, the attribution assigned to feature j is proportional to the gradient of the model with respect to that feature:

$$\phi_j^{\text{SIGMA}} \propto \frac{\partial f(\boldsymbol{\sigma}_k)}{\partial \sigma_k^{(j)}}. \tag{6}$$

Since j is dummy, we have:

$$\frac{\partial f(\boldsymbol{\sigma}_k)}{\partial \sigma_k^{(j)}} = 0 \quad \text{for all } k. \tag{7}$$

Thus, the attribution $\phi_j^{\text{SIGMA}} = 0$, and the Sensitivity(b) axiom is satisfied.

Definition 3 (Implementation Invariance)**.** *The implementation invariance axiom states that if two networks are functionally equivalent, that is, they produce the same outputs for all inputs, then their attributions should also be identical. The attribution should not rely on the implementation details of the network.*

Remark 2. *SIGMA satisfies Implementation Invariance because its attribution paths depend solely on the output of the network and the gradients of that output. Consequently, if two networks yield the same output for all inputs, they will generate identical attribution paths in response to the same random perturbations. As discussed in [30], this reliance on output-level gradients ensures that SIGMA is implementation invariant.*

Definition 4 (Symmetry)**.** *As discussed in [31], a function is symmetric if, for two variables i and j, it satisfies $f(\ldots, \sigma_k^i, \sigma_k^j, \ldots) = f(\ldots, \sigma_k^j, \sigma_k^i, \ldots)$. Consequently, the attributions for i and j under a symmetric function should be equal if $\sigma_k^i = \sigma_k^j$ at each point in the integration path, k.*

Remark 3. *SIGMA preserves symmetry if, at every iteration, symmetric features $\sigma_k^i = \sigma_k^j$ receive identical perturbations along the SIGMA path. Under this condition, their values remain equal at each step σ_k, and the resulting attributions will be identical. We formalise this claim in the following proof.*

Proof. Let $\boldsymbol{\sigma}_{k=0}$ be the input to be explained, and suppose features i and j are symmetric with $\sigma_{k=0}^{(i)} = \sigma_{k=0}^{(j)}$. Assume that at each iteration, these features receive identical perturbations: $\Delta_k^{(i)} = \Delta_k^{(j)}$ for all k.

According to the update rule used in SIGMA, the update to $\boldsymbol{\sigma}_k$ at each iteration, defining the SIGMA path, is given in Eq.3 and expanded using Eq.2:

$$\boldsymbol{\sigma}_{k+1} = \boldsymbol{\sigma}_k - \alpha \frac{f(\boldsymbol{\sigma}_k + \beta\boldsymbol{\Delta}_k) - f(\boldsymbol{\sigma}_k - \beta\boldsymbol{\Delta}_k)}{2\beta} \boldsymbol{\Delta}_k, \quad (8)$$

The constants α and β are uniform across all features. Since f is symmetric in variables i and j, and if we assume the perturbations $\Delta_k^{(i)}$ and $\Delta_k^{(j)}$ to be equal, it follows that

$$f(\sigma_k^{(i)} \pm \beta\Delta_k^{(i)}) = f(\sigma_k^{(j)} \pm \beta\Delta_k^{(j)}). \quad (9)$$

Hence, the updates to each feature are equal: $\sigma_{k+1}^i = \sigma_{k+1}^j$. The equation for the attribution for each input feature is given as,

$$\phi_i^{\text{SIGMA}} = \sum_{k=1}^{N} g_k^{(i)} \times \frac{f(\boldsymbol{\sigma}_{k-1}) - f(\boldsymbol{\sigma}_k)}{\sum_{i'} g_k^{(i')}} \quad (10)$$

Where, $g_k^{(i)} = \frac{\partial f(\boldsymbol{\sigma}_k)}{\partial \sigma_k^{(i)}}$, is the model gradient for feature i. Lemma 1 in [30] proves that the partial derivatives of a symmetric function are equal if the values of the symmetric variables are equal. This implies $g_k^{(i)} = g_k^{(j)}$.

Further, we show that the weighting term used in the gradient accumulation continues to preserve symmetry. By the definition of a symmetric function:

$$f(\sigma_k^{(i)}) = f(\sigma_k^{(j)}) \quad \text{for all } k. \quad (11)$$

This ensures that by definition, the drop in model confidence at each stage in the path is identical for features i and j. The sum over all gradients in the gradient map is then used for normalisation, $\sum_{i'} g_k^{(i')}$, and is independent of the current feature we are considering, remaining equal for both i and j.

We have shown that, under the assumption of identical perturbations, symmetric features i and j are equal at every point k on the path $\sigma_k^{(i)} = \sigma_k^{(j)}$. They then receive identical model gradients $g_k^{(i)} = g_k^{(j)}$ that are not affected by the weighting term. Therefore we prove that SIGMA satisfies symmetry.

A3. Effect of Perturbation Pattern on Final Attribution

In line with the original SPSA algorithm, the perturbations made to the input $\boldsymbol{\Delta}_k$ are drawn from the symmetric Bernoulli distribution and subsequently scaled to match the image range expected by the model. Here we investigate the use of various noise patterns to perturb the input image. This is done over RGB channels for 3 channel input images but remains the same for single channel, greyscale images. Figure 11(a) visualises the average attribution map after 7 runs for Bernoulli, Gaussian and Perlin noise patterns. Figure 11(b) makes use of the ability to easily change scale with Perlin noise to explore the effect of perturbation scale on the final attribution. The attribution maps all highlight the appropriate areas with the smaller scales providing the most spatially coherent attributions, thus suggesting the specific noise pattern does not affect conclusions made using the SIGMA approach. To provide a fair comparison all step size and perturbation magnitudes were kept constant. However, at the larger length scales, with more of the image being obscured in a given iteration, different perturbation parameters would provide a slower decay of model confidence, resulting in a smoother and more coherent attribution.

A4. Parameter Study

A4.1. Stopping Threshold (ϵ)

As SIGMA follows the collapse of model confidence, it does not have a pre-defined end point to the path or number of integration steps. Instead, it dynamically terminates when the model's confidence drops below 1%, referred to as $\epsilon = 0.01$ in the pseudocode. This value was chosen as it encompasses the full path without leading to excessive computation times with diminishing returns. Figure 13 demonstrates the increase in attribution quality that comes from

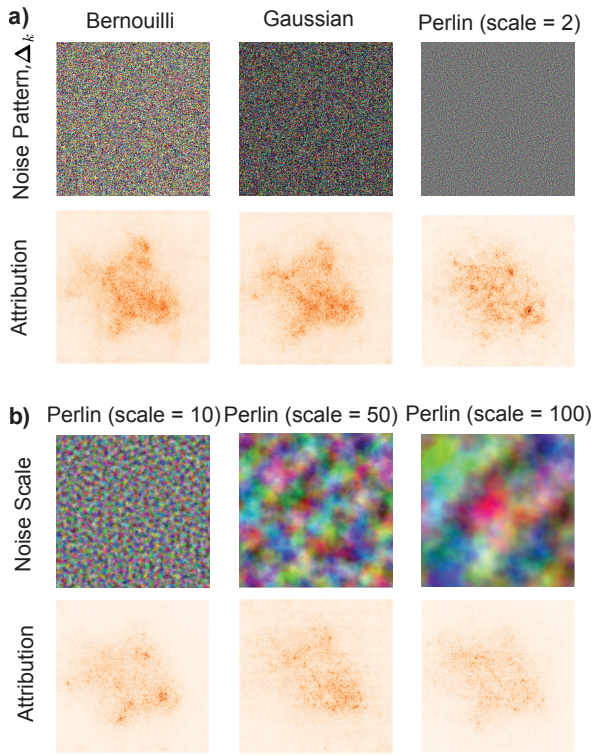

Figure 11. Average attribution maps ($n = 7$) for different noise patterns (a) and scales (b) used with the 'Goldfish' example from Figure 1.

allowing the confidence curve to fully decay and highlights the computational expense of letting it continue to values lower than 1%.

A4.2. Perturbation Magnitude (β)

The perturbation magnitude β determines the scale of the random perturbation Δ_k applied to the input at each iteration. Fig. 12 (a) depicts the resulting SIGMA path trajectories for various values of β, while Fig. 12 (c) presents the corresponding attribution maps. When β is chosen to be excessively large (e.g. $\beta = 1$), the model's confidence on the perturbed inputs collapses to near zero after the first iteration, causing the gradient $\nabla J(\sigma)$ between successive perturbed samples to vanish. Consequently, unless the step size α is scaled proportionally, the attribution path fails to advance, as the negligible gradient between perturbed images prevents effective updates to the image. When working with normalised confidences such as softmax, it is found that perturbation magnitudes of $0.01 \leq \beta \leq 0.5$ lead to consistent attribution maps, with the main trade-off being computation time as smaller perturbations take longer for the model's confidence to be minimised.

A4.3. Step Size (α)

The step size, α, determines how far we move along the direction given by the gradient, $\nabla J(\sigma_k)$. Although $\nabla J(\sigma_k)$ provides both magnitude and direction, relying directly on its magnitude can result in steps that are either too large, hindering convergence, or too small, leading to slow progress. Incorporating the step size parameter α allows appropriate scaling of the gradient, ensuring stable and efficient convergence while preserving the information obtained from perturbations.

Figure. 12(b) illustrates the effect of varying α on the SIGMA path while keeping the perturbation size fixed at $\beta = 0.1$. The step size α primarily influences convergence time, with larger values leading to faster convergence. However, excessively large steps can cause the model's confidence in the perturbed image to drop too quickly, resulting in gradient saturation similar to choosing a large β. To mitigate this, α should be chosen such that the model's confidence decreases gradually over several iterations, ensuring stable gradients and more reliable attribution maps.

As shown in Fig. 12(c), attribution maps for $\alpha \leq 1$ and β within the range $0.01 \leq \beta \leq 0.5$ yield stable results, this was found to be true across the networks and images tested in section 4.

A5. Path Averaging

Due to the stochastic nature of the SIGMA path, the exact trajectory from the original image to the maximally perturbed image varies across iterations. However, because the perturbations are small, the resulting attribution maps exhibit minimal variability and remain consistent across runs. Fig.14(a) illustrates 100 SIGMA paths for a single image, with the average path shown in bold and all individual paths contained within the shaded region. This variability enables averaging over multiple paths to improve robustness. Fig.14(b) presents the resulting attribution maps obtained by averaging over n paths. While all attributions consistently highlight the same region of the image, increasing n results in less sparse maps. However, this comes at the cost of increased computation time and diminishing improvements in attribution quality. It was found that averaging over $n = 7$ paths yields clear and stable attributions, while maintaining a similar computation time (27 seconds) to IG for a path with 200 points (42 seconds). Please see SectionA6 for a full discussion. Overlaying the resulting $n = 7$ path in Fig.14(a) (orange dashed line), shows little deviation from the $n = 100$ path but the computation time is ≈12x longer. In addition to averaging, employing multiple stochastic paths enables the generation of a confidence map alongside the attribution map. This confidence map reflects the method's certainty in its attributions and is computed by taking the standard deviation across the set of attribution

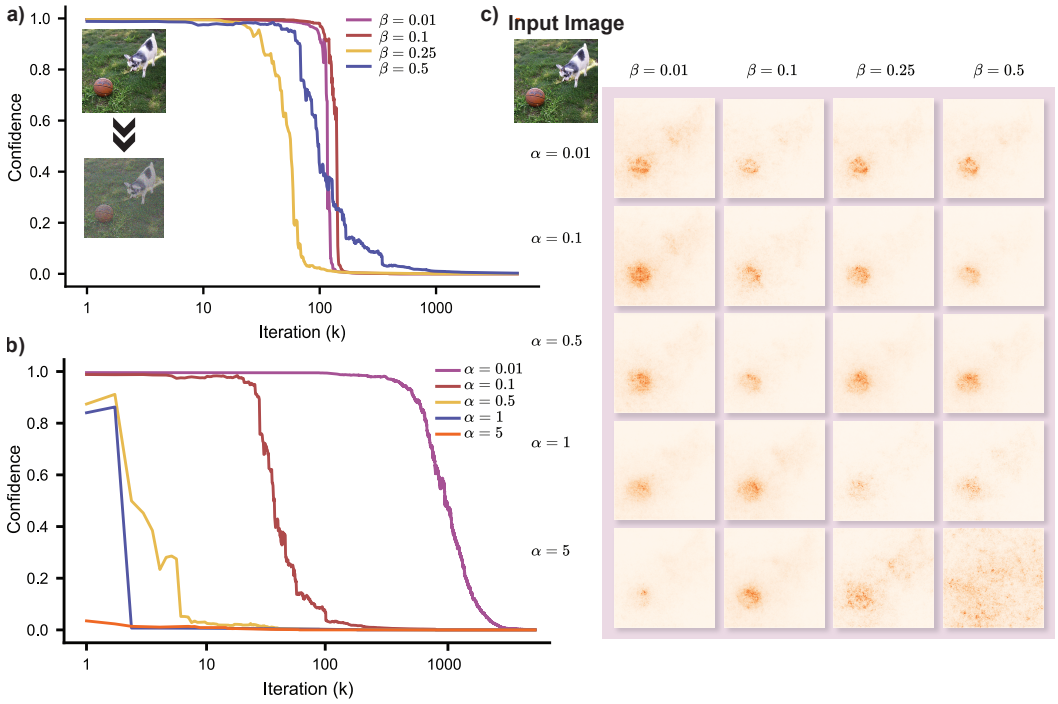

Figure 12. The path taken for the explanation of class 'Basketball' for various values of **(a)** perturbation magnitude β, with constant $\alpha = 0.1$ and **(b)** step size α, with constant $\beta = 0.5$. The initial image to be explained and resulting maximally perturbed image are shown inset. The robustness of the final attribution to changes in α and β is demonstrated in **(c)** with the exception of large values that lead to saturated paths.

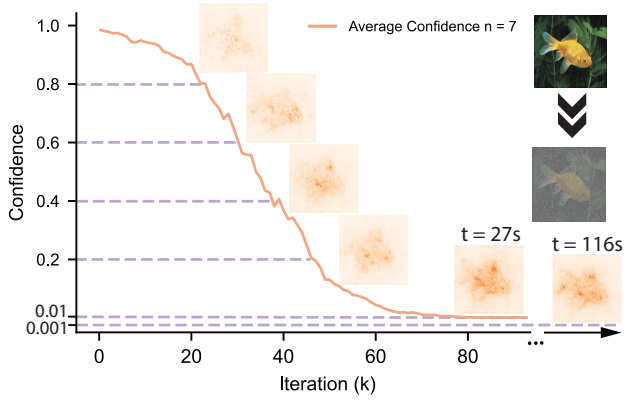

Figure 13. Effect of terminating the SIGMA algorithm at various stages along the path. Stopping before the confidence has reached near-zero results in sparse, incomplete attributions. Setting the stopping threshold too low leads to excessive computation times with diminishing returns.

maps. This measure offers a valuable means of quantifying the reliability of the attributions, highlighting regions where the model's explanations are more or less stable. For each of the results presented in this work, the accompanying confidence maps can be found in the supplementary.

A6. Computational Cost

We analyse the computational cost of SIGMA in comparison to other attribution methods evaluated in this study. Table 3 reports the average time required to generate a single attribution map for each method across different datasets. The total runtime of SIGMA depends on the number of stochastic paths employed, see Fig 15. For direct comparison we report one path for each method below, as IG and GIG do not support path averaging in their standard form. In practice, all SIGMA paths are independent and can therefore be computed in parallel, substantially reducing computation time on modern GPUs. All analysis in this work was carried out on a NVIDIA GeForce RTX 3090 GPU. As in Section 4 the SIGMA algorithm runs until terminated when model confidence drops below 1% ($\epsilon = 0.01$) and each other method had a predefined number of steps set to 200, in line with the recommendations in literature [25].

A7. Quantitative Benchmarks to Measure Human Alignment

In Section 4 we use two metrics outlined in Saliency-Bench [39], the mean Intersection over Union (mIoU) and the Pointing Game (PG). These are two common alignment metrics that evaluate how well the attribution map aligns

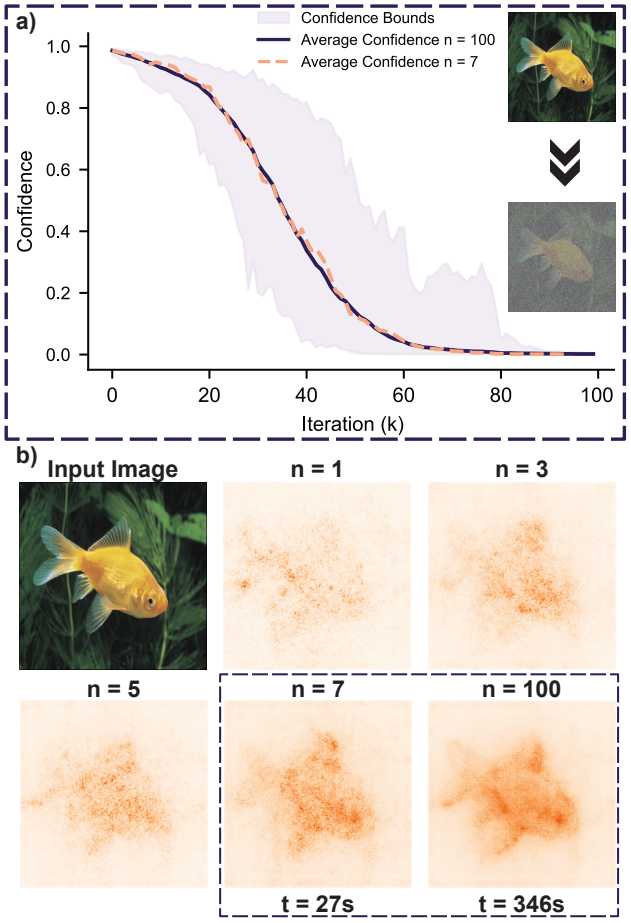

Figure 14. Effect of averaging over multiple runs with parameters, $\alpha = 0.1$ and $\beta = 0.5$. The stochastic nature of SPSA results in slightly different paths taken for each run. (a) Average path taken over 100 different runs (solid line) with maximum variation in runs shown in shading and the average path over 7 runs overlaid (dashed line). The initial image to be explained and resulting maximally perturbed image are shown inset. (b) Attribution maps for averaging over different numbers of runs (n). It should be noted that after 7 runs diminishing returns can be seen in the attribution map with respect to the rising computational cost (t) given in seconds.

Table 3. Average computation time (seconds per input) required to generate an attribution map across 1000 images in each dataset.

Method	ImageNet	Disease-XAI	Security-XAI
IG	5.25	0.37	2.49
GIG	81.69	4.86	42.85
IG^2	20.52	9.72	91.30
SIGMA	39.75	8.02	15.78

with the human-annotated ground truth explanation. We

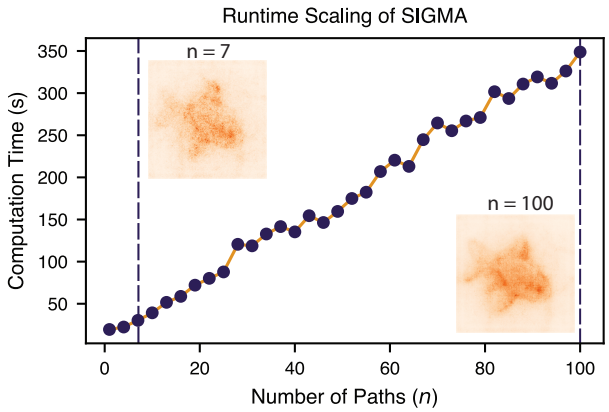

Figure 15. SIGMA computation time as a function of path number. There is a linear increase in computation time, with diminishing returns in attribution quality beyond $n = 7$ paths; see also Fig. 14.

outline formal definitions for each below:
The mIoU metric is calculated by introducing a threshold θ to convert the attribution map to a binary explanation map B. The following formula is used to compare each binary map B_j to the corresponding binary human annotation A_j, averaged over M attribution maps obtained across the dataset,

$$\text{mIoU}(\phi^{SIGMA}, A) = \frac{1}{M}\sum_{j=1}^{M}\frac{|B_j \cap A_j|}{|B_j \cup A_j|}. \quad (12)$$

The PG aims to quantify alignment by determining if the peak of the attribution map lies within the human-annotated explanation region. In this case the binary map is not needed and the original attribution map is used. The formula for PG is as follows,

$$\text{Pointing Game} = \frac{\sum_{j=1}^{M} 1[\text{MaxLoc}(\phi_j^{SIGMA}) \in A_j]}{N}. \quad (13)$$

These alignment metrics were used alongside the faithfulness metric, SIC-AUC to evaluate SIGMA on two of the Saliency-Bench datasets, Disease-XAI and Security-XAI.