Understanding Impacts of High-Order Loss Approximations and Features in Deep Learning Interpretation

Sahil Singla [1] **Eric Wallace** [1] **Shi Feng** [1] **Soheil Feizi** [1]

Abstract

Current methods to interpret deep learning models by generating saliency maps generally rely on two key assumptions. First, they use first-order approximations of the loss function neglecting higher-order terms such as the loss curvatures. Second, they evaluate each feature's importance in isolation, ignoring their inter-dependencies. In this work, we study the effect of relaxing these two assumptions. First, by characterizing a closed-form formula for the Hessian matrix of a deep ReLU network, we prove that, for a classification problem with a large number of classes, if an input has a high confidence classification score, the inclusion of the Hessian term has small impacts in the final solution. We prove this result by showing that in this case the Hessian matrix is approximately of rank one and its leading eigenvector is almost parallel to the gradient of the loss function. Our empirical experiments on ImageNet samples are consistent with our theory. This result can have implications in other related problems such as adversarial examples as well. Second, we compute the importance of group-features in deep learning interpretation by introducing a sparsity regularization term. We use the $L_0 - L_1$ relaxation technique along with the proximal gradient descent to have an efficient computation of group feature importance scores. Our empirical results indicate that considering group features can improve deep learning interpretation significantly.

1. Introduction

The growing use of deep learning in sensitive applications such as medicine, autonomous driving, and finance raises concerns about human trust in machine learning systems. For trained models, a central question is test-time *interpretability*: how can humans understand the reasoning behind model predictions? A common interpretation approach is to identify the importance of input features for a model's prediction. A saliency map can then visualize important pixels of an image (Simonyan et al., 2014; Sundararajan et al., 2017) or words in a sentence (Li et al., 2016).

In the last couple of years, several approaches have been proposed to tackle this problem. For example, reference (Simonyan et al., 2014) computes the gradient of the class score with respect to the input while reference (Smilkov et al., 2017) computes the average gradient-based importance values generated from several noisy versions of the input. Reference (Sundararajan et al., 2017) defines a *baseline*, which represents an input absent of information and determines feature importance by accumulating gradient information along the path from the baseline to the original input. Reference (Alvarez-Melis & Jaakkola, 2018) builds interpretable neural networks by learning basis concepts that satisfy an interpretability criteria while reference (Adebayo et al., 2018) proposes methods to assess the scope and quality of saliency maps. Although these methods can produce visually pleasing results, they often make weak model approximations (Adebayo et al., 2018; Nie et al., 2018) and can be sensitive to noise and adversarial perturbations (Ghorbani et al., 2017; Kindermans et al., 2017).

Existing deep learning interpretation methods mainly rely on two key assumptions:

- **The gradient-based loss surrogate assumption:** For computational efficiency, several existing methods (e.g. (Simonyan et al., 2014; Smilkov et al., 2017; Sundararajan et al., 2017)) assume that the loss function is almost linear at the test point. Thus, they use variations of the input gradient to compute feature importance.
- **The isolated feature importance assumption:** Current methods evaluate the importance of each feature in isolation, assuming all other features are fixed. Features, however, may have complex inter-dependencies that can be learned by the model.

In this work, we study the impact of relaxing these assump-

[1]Computer Science Department, University of Maryland. Correspondence to: Sahil Singla <ssingla@cs.umd.edu>, Soheil Feizi <sfeizi@cs.umd.edu>.

tions in deep learning interpretation. To relax the first assumption, we use the second-order approximation of the loss function by keeping the Hessian term in its Taylor's expansion. For a deep ReLU network and the cross entropy loss function, we compute the Hessian term in *closed-form*. Using the closed-form formula for the Hessian matrix, we prove the following result:

Theorem 1 (informal version) *If the probability of the predicted class is close to one and the number of classes is large, solutions of the first-order and second-order interpretation methods are sufficiently close to each other.*

We present a formal version of this result in Theorem 5. We validate this result empirically as well. For example, in ImageNet which has more than 1,000 classes, we show that incorporating the Hessian term in deep learning interpretation has small impact for most images. This is consistent with our theory.

The key proof idea of this result follows from the fact that when the number of classes is large and the confidence in the predicted class is high, the Hessian of the loss function is approximately of rank one. More specifically, the largest eigenvalue squared is significantly larger than the sum of squared remaining eigenvalues. Moreover, the corresponding eigenvector is approximately parallel to the gradient vector (Theorem 4). This makes the first-order and second-order methods to perform similarly to each other. Note that this result can be extended to some other related problems such as adversarial examples where most common methods are based on the first-order approximation of the loss function(Carlini & Wagner, 2016; Goodfellow et al., 2014; Moosavi-Dezfooli et al., 2015).

In the second part of the paper, we relax the second assumption of current interpretation approaches (i.e. the isolated feature importance assumption). To incorporate feature inter-dependencies in deep learning interpretation, we define the importance function over *subsets* of features, referred to as *group-features*. We adjust the subset size on a per-example basis using an unsupervised approach, making the interpretation method *context-aware*. Including the group-feature in deep learning interpretation makes the optimization to be combinatorial. To circumvent computational issues, we use an $L_0 - L_1$ relaxation as is common in compressive sensing (Candes & Tao, 2005; Donoho, 2006), the LASSO regression (Tibshirani, 1996), etc. To efficiently compute a solution for the relaxed optimization, we employ the proximal gradient descent (Parikh & Boyd, 2014). Our empirical results indicate that incorporating group-features significantly improves the quality of interpretation results.

Below we summarize our contributions in this paper:

- We study the impact of the second-order approximation of the loss function in deep learning interpretation. We prove that, under certain conditions, solutions of the first-order and second-order interpretation methods are sufficiently close to each other (Theorems 4 and 5). This result can be insightful in other related problems such as adversarial examples. Our empirical results on ImageNet samples are consistent with our theory.
- To prove the above Hessian result, we compute the Hessian matrix of a deep ReLU network in a closed-form. This result can be of independent interest to readers (Proposition 1 and Theorem 2).
- Finally, we include inter-dependencies among features in deep learning interpretation by computing the importance of *group-features*. Borrowing some results from compressive sensing (Section 4 and Appendix Section E), we develop a computationally efficient approach to solve the underlying optimization. Our empirical results indicate that considering group features significantly improves deep learning interpretation.

In what follows, we explain these results in more details. All proofs have been presented in Supplementary Materials.

2. Problem Setup and Notation

Consider a prediction problem from input variables (features) $X \in \mathcal{X} \subset \mathbf{R}^d$ to an output variable $Y \in \mathcal{Y}$. For example, in the image classification problem, $\mathcal{X}$ is the space of images and $\mathcal{Y}$ is the set of labels $\{1, ..., c\}$. We observe m samples from these variables, namely $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m)\}$. Let $\mathbb{P}_{X,Y}$ be the observed empirical distribution.[1] The empirical risk minimization (ERM) approach computes the optimal predictor $f_\theta : \mathcal{X} \to \mathcal{Y}$ for a loss function $\ell(\cdot, \cdot)$ using the following optimization:

$$\min_{\theta \in \Theta} \; \mathbb{E}_{\mathbb{P}_{X,Y}} \left[\ell \left(f_\theta(\mathbf{x}), y \right) \right]. \tag{1}$$

Let $\mathcal{S}$ be a subset of $[d] := \{1, 2, ..., d\}$ with cardinality $|\mathcal{S}|$. For a given sample $(\mathbf{x}, y)$, let $\mathbf{x}(\mathcal{S})$ indicate the features of $\mathbf{x}$ in positions $\mathcal{S}$. We refer to $\mathbf{x}(\mathcal{S})$ as a group-feature of $\mathbf{x}$. The importance of a group-feature $\mathbf{x}(\mathcal{S})$ is proportional to the change in the loss function when $\mathbf{x}(\mathcal{S})$ is perturbed. We select the group-feature with maximum importance and visualize that subset in a saliency map.

Definition 1 (Group-Feature Importance Function) *Let θ^* be the optimizer of the ERM problem (1). For a given sample $(\mathbf{x}, y)$, we define the group-feature importance*

[1] Note that for simplicity, we hide the dependency of $\mathbb{P}_{X,Y}$ on m.

function $I^{k,\rho}_{\theta^*}(\mathbf{x}, y)$ as follows:

$$I^{k,\rho}_{\theta^*}(\mathbf{x}, y) \coloneqq \max_{\tilde{\mathbf{x}}} \ \ell\left(f_{\theta^*}(\tilde{\mathbf{x}}), y\right) \quad \|\tilde{\mathbf{x}} - \mathbf{x}\|_0 \le k, \quad \|\tilde{\mathbf{x}} - \mathbf{x}\|_2 \le \rho, \tag{2}$$

where $\|.\|_0$ counts the number of non-zero elements of its argument (known as the L_0 norm). The parameter k characterizes an upper bound on the cardinality of the group-features. The parameter ρ characterizes an upper bound on the L_2 norm of feature perturbations.

If $\tilde{\mathbf{x}}^*$ is the solution of optimization (2), then the vector $|\tilde{\mathbf{x}}^* - \mathbf{x}|$ is the feature importance values that are visualized in the saliency map. Note, when $k = 1$ this definition simplifies to current feature importance formulations which consider features in isolation. When $k > 1$, our formulation can capture feature interdependencies. Parameters k and ρ in general depend on the test sample $\mathbf{x}$ (i.e., the size of the group-features are different for each image and model). We introduce an unsupervised metric to determine these parameters in Section 4.1, but assume these parameters are given for the time being.

The cardinality constraint $\|\tilde{\mathbf{x}} - \mathbf{x}\|_0 \le k$ (i.e. the constraint on the group-feature size) leads to a combinatorial optimization problem in general. Such a sparsity constraint has appeared in different problems such as compressive sensing (Candes & Tao, 2005; Donoho, 2006) and LASSO regression (Tibshirani, 1996). Under certain conditions, one can show that without loss of generality, the L_0 norm can be relaxed with the (convex) L_1 norm (Appendix Section E).

Our goal is to solve optimization (2) which is non-linear and non-concave in $\tilde{\mathbf{x}}$. Current approaches do not consider the cardinality constraint and optimize $\tilde{\mathbf{x}}$ by linearizing the objective function (i.e., using the gradient). To incorporate group features into the current methods, we can add the constraints of optimization (2) to the objective function using Lagrange multipliers. This yields the following Context-Aware First-Order (CAFO) interpretation function.

Definition 2 (The CAFO Interpretation) *For a given sample $(\mathbf{x}, y)$, we define the Context-Aware First-Order (CAFO) importance function $\tilde{I}^{\lambda_1,\lambda_2}_{\theta^*}(\mathbf{x}, y)$ as follows:*

$$\tilde{I}^{\lambda_1,\lambda_2}_{\theta^*}(\mathbf{x}, y) \coloneqq \max_{\Delta} \ \nabla_{\mathbf{x}} \ell\left(f_{\theta^*}(\mathbf{x}), y\right)^t \Delta - \lambda_1 \|\Delta\|_1 - \lambda_2 \|\Delta\|_2^2 \tag{3}$$

where λ_1 and λ_2 are non-negative regularization parameters. We refer to the objective of this optimization as $\tilde{\ell}(\Delta)$, hiding its dependency on $(\mathbf{x}, y)$ and θ^* to simplify notation.

Large values of regularization parameters λ_1 and λ_2 in optimization (3) correspond to small values of parameters k and ρ in optimization (2). Incorporating group-features naturally leads to a sparsity regularizer through the L_1 penalty. Note, this is not a hard constraint which forces a sparse interpretation. Instead, given proper choice of the regularization coefficients, the interpretation will reflect the sparsity used by the underlying model. In Section 4.1, we detail our method for setting λ_1 for a given test sample (context-aware) based on the sparsity ratio of CAFO's optimal solution. Moreover, in Appendix Section E, we show that under some general conditions, optimization (3) can be solved efficiently and its solution matches that of the original optimization (2).

To have a better approximation of the loss function, we use the second-order Taylor expansion of the loss function around point $(\mathbf{x}, y)$ as follows:

$$\ell\left(f_{\theta^*}(\tilde{\mathbf{x}}), y\right) \approx \ell\left(f_{\theta^*}(\mathbf{x}), y\right) + \underbrace{\nabla_{\mathbf{x}} \ell\left(f_{\theta^*}(\mathbf{x}), y\right)^t \Delta}_{\text{the first-order term}} + \underbrace{\frac{1}{2} \Delta^t \mathbf{H}_{\mathbf{x}} \Delta}_{\text{the second-order term}} \tag{4}$$

where $\Delta \coloneqq \tilde{\mathbf{x}} - \mathbf{x}$ and $\mathbf{H}_{\mathbf{x}}$ is the Hessian of the loss function on the input features $\mathbf{x}$ (note y is fixed). This second-order expansion of the loss function decreases the interpretation's model approximation error.

We show that by choosing proper values for regularization parameters, the resulting optimization using the second-order surrogate loss is strictly a convex minimization (or equivalently concave maximization) problem, allowing for efficient optimization using gradient descent (Theorem 3). Moreover, even though the Hessian matrix $\mathbf{H}_{\mathbf{x}}$ can be expensive to compute for large neural networks, gradient updates of our method only require the Hessian-vector product (i.e., $\mathbf{H}_{\mathbf{x}}\Delta$) which can be computed efficiently (Pearlmutter, 1994). This yields the following Context-Aware Second-Order (CASO) interpretation function.

Definition 3 (The CASO Interpretation) *For a given sample $(\mathbf{x}, y)$, we define the Context-Aware Second-Order (CASO) importance function $\tilde{I}^{\lambda_1,\lambda_2}_{\theta^*}(\mathbf{x}, y)$ as follows:*

$$\tilde{I}^{\lambda_1,\lambda_2}_{\theta^*}(\mathbf{x}, y) \coloneqq \max_{\Delta} \ \nabla_{\mathbf{x}} \ell\left(f_{\theta^*}(\mathbf{x}), y\right)^t \Delta + \frac{1}{2} \Delta^t \mathbf{H}_{\mathbf{x}} \Delta - \lambda_1 \|\Delta\|_1 - \lambda_2 \|\Delta\|_2^2 \tag{5}$$

We refer to the objective of this optimization as $\tilde{\ell}(\Delta)$. λ_1 and λ_2 are defined as in (3).

3. Understanding Impact of the Hessian in Deep Learning Interpretation

Hessian is by definition useful when the loss function at the test point has high curvature. However, given the linear na-

ture of popular network architectures with piecewise linear activations (e.g., ReLu (Glorot et al., 2011), Maxout (Goodfellow et al., 2013)), *do these regions of high curvature even exist?* We answer this question for neural networks with piecewise linear activations by providing an exact calculation of the input Hessian. We use this derivation to understand the impact of including the Hessian term in deep learning interpretation. More specifically, we prove that when the probability of the predicted class is ≈ 1 and the number of classes is large, the second-order interpretation is similar to the first-order one. We verify this theoretical result experimentally over images in the IMAGENET dataset. We also observe that when the confidence in the predicted class is low, the second-order interpretation can be significantly different from the first-order interpretation. Since second-order interpretations take into account the curvature of the model, we conjecture that they are more faithful to the underlying model in these cases.

3.1. A Closed-form Hessian Formula for Deep ReLU Networks

We present an abridged version of the exact Hessian calculation here while the details are provided in Appendix Section A.1. Neural network models which use piecewise linear activation functions have class scores (logits) which are linear functions of the input [2]. The network can thus be written as:

$$f_\theta(\mathbf{x}) = \mathbf{W}^T\mathbf{x} + \mathbf{b},$$

where $\mathbf{x}$ is the input of dimension d, $f_\theta(\mathbf{x})$ are the logits, $\mathbf{W}$ are the weights, and $\mathbf{b}$ are the biases of the linear function. Note that $\mathbf{W}$ combines weights of different layers from the input to the output of the network. Each row $\mathbf{W}_i$ of $\mathbf{W}$ is the gradient of logit $f_\theta(\mathbf{x})_i$ with respect to flattened input $\mathbf{x}$ and can be handled in auto-grad software such as PyTorch (Paszke et al., 2017). We define:

$$\mathbf{p} = \text{softmax}(f_\theta(\mathbf{x}))$$
$$\ell(f_\theta(\mathbf{x}), y) = -\sum_{i=1}^{c} y_i \log(\mathbf{p}_i),$$

where c denotes the number of classes, $\mathbf{p}$ denotes the class probabilities, and $\ell(\mathbf{p}, \mathbf{y})$ is the cross-entropy loss function.

In this case, we have the following result:

Proposition 1 $\mathbf{H}_\mathbf{x}$ *is given by:*

$$\mathbf{H}_\mathbf{x} = \nabla^2_\mathbf{x}\ell(\mathbf{p}, \mathbf{y}) = \mathbf{W}(diag(\mathbf{p}) - \mathbf{p}\mathbf{p}^T)\mathbf{W}^T \quad (6)$$

where $\text{diag}(\mathbf{p})$ is a diagonal matrix whose diagonal elements are equal to $\mathbf{p}$.

[2]Note that we ignore points that the function is non-differentiable at as they form a measure zero set.

The first observation from Proposition 1 is as follows:

Theorem 2 $\mathbf{H}_\mathbf{x}$ *is a positive semidefinite matrix.*

These two results allow an extremely efficient computation of the Hessian's eigenvectors and eigenvalues using the Cholesky decomposition of $\text{diag}(\mathbf{p}) - \mathbf{p}\mathbf{p}^T$. See Appendix Section C for full details. Note the use of decomposition is critical as storing the Hessian requires intractable amounts of memory for high dimensional inputs. The entire calculation of the Hessian's decomposition for ImageNet using a ResNet-50 (He et al., 2016) runs in approximately 4.2 seconds on an NVIDIA GTX 1080 Ti.

To the best of our knowledge, this is the first work which derives the exact Hessian decomposition for piecewise linear networks. Yao et al. 2018 (Yao et al., 2018) also proved the Hessian for piecewise linear networks is at most rank c but did not derive the exact input Hessian.

One advantage of having a closed-form formula for the Hessian matrix (6) is that we can use it to properly set the regularization parameter λ_2 in CASO's formulation. To do this, we rely on the following result:

Theorem 3 *If L is the largest eigenvalue of $\mathbf{H}_\mathbf{x}$, for any value of $\lambda_2 > L/2$, the second-order interpretation objective function (5) is strongly concave.*

We use Theorem 3 to set the regularization parameter λ_2 for CASO. We need to set λ_2 to make the optimization convex, but not set λ_2 so large that it overpowers $\mathbf{H}_\mathbf{x}$. In particular, we set $\lambda_2 = L/2 + c_1$, where we choose $c_1 = 10$. For CAFO, we set $\lambda_2 = c_1 = 10$.. We estimate L using the power-iteration method. In our experiments, we found that around 10 iterations are sufficient for convergence of the power iteration method.

3.2. Theoretical results on the Hessian impact

In this section, we leverage the exact Hessian calculation to prove that when the probability of predicted class is ≈ 1 and number of classes is large, the Hessian of a piecewise linear neural network is approximately of rank one and its eigenvector is approximately parallel to the gradient. Since a constant scaling does not affect the visualization, this causes the two interpretations to be approximately similar to each other.

Theorem 4 *If the probability of the predicted class=1-(c-1)ϵ , where $\epsilon \approx 0$, then as $c \to \infty$ such that $c\epsilon \approx 0$, Hessian is of rank one and its eigenvector is parallel to the gradient.*

Let Δ^*_{CASO} be the optimal solution to the CASO objective 5 and Δ^*_{CAFO} be the optimal solution for the CAFO objective 3. We assume λ_1=0 for both the objectives.

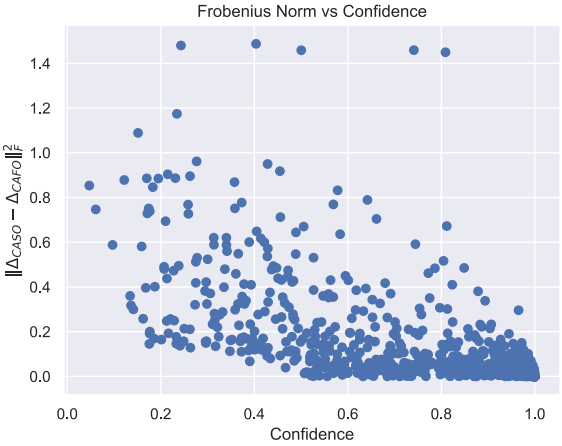


Figure 1. Scatter plot showing the Frobenius norm difference between CASO and CAFO (after normalizing both vectors to have the same L_2 norm). Consistent with the result of Theorem 4, when the classification confidence probability is low, the CASO result differs significantly than that of CAFO. When confidence is high, CASO and CAFO lead approximately to the same interpretation. To isolate the impact of the Hessian term, we assume $\lambda_1 = 0$ in both CASO and CAFO optimizations.

Theorem 5 *If the probability of the predicted class=1-(c-1)ϵ , where $\epsilon \approx 0$, then as $c \to \infty$ such that $c\epsilon \approx 0$, the CASO solution (5) with $\lambda_1 = 0$ is almost parallel to the CAFO solution (3) with $\lambda_1 = 0$.*

3.3. Empirical results on the Hessian impact

In this section, we present empirical results on the impact of the second-order loss approximation in deep learning interpretation. In experiments of this section, to isolate the impact of the Hessian term, we assume $\lambda_1 = 0$ in both CASO and CAFO optimizations.

A consequence of Theorem 3 is that the gradient descent method with Nesterov momentum converges to the global optimizer of the second-order interpretation objective objective with a convergence rate of $\mathcal{O}(1/t^2)$ (Appendix Section B).

To optimize Δ, the gradient is given by:

$$\nabla_\Delta \tilde{\ell}(\Delta) = \nabla_{\mathbf{x}} \ell\left(f_{\theta^*}(\mathbf{x}), y\right) + \mathbf{H}_{\mathbf{x}}\Delta - 2\lambda_2\Delta. \quad (7)$$

The gradient term $\nabla_{\mathbf{x}} \ell\left(f_{\theta^*}(\mathbf{x}), y\right)$ and the regularization term $-2\lambda_2\Delta$ are straightforward to implement using standard backpropagation.

To compute the Hessian-vector product term $\mathbf{H}_{\mathbf{x}}\Delta$, we rely on the result of Pearlmutter 1994 (Pearlmutter, 1994): a Hessian-vector product can be computed in the same time as the gradient $\nabla_{\mathbf{x}} \ell\left(f_{\theta^*}(\mathbf{x}), y\right)$. This is handled easily in modern auto-grad software. Moreover, for ReLU networks, our closed-form formula for the Hessian term (Theorem 1) can be used in computation of the Hessian-vector product as well. In our experiments, we use the closed-form formula for $\lambda_1 = 0$ and proximal gradient descent for $\lambda_1 > 0$.

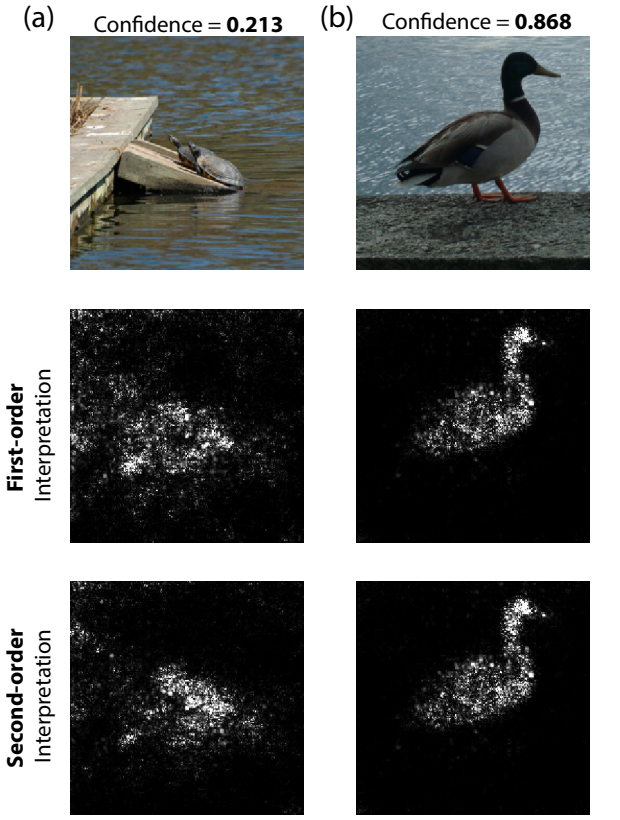


Figure 2. Panel (a) shows an example where the classification confidence is low. In this case, the CASO and CAFO interpretations differ significantly. Panel (b) demonstrates an example where the classification confidence is high. In this case, CASO and CAFO lead to similar interpretations.

We compare second-order interpretations (CASO with $\lambda_1 = 0$) and first-order variant (CAFO with $\lambda_1 = 0$) empirically. Note that when $\lambda_1 = 0$, $\Delta_{CAFO} = \frac{1}{\lambda_2}\mathbf{g}_{\mathbf{x}}$ where $\mathbf{g}_{\mathbf{x}}$ is the gradient and Δ_{CAFO} is the interpretation obtained using the CAFO objective.

We compute second-order and first-order interpretations for 1000 random samples on the ImageNet ILSVRC-2013 (Russakovsky et al., 2015) validation set using a Resnet-50 (He et al., 2016) model. Our loss function $\ell(\cdot,\cdot)$ is the cross-entropy loss. After calculating Δ for all methods, the values must be normalized for visualization in a saliency map. We apply a normalization technique from existing work which we describe in Appendix Section D.

We plot the Frobenius norm of the difference between CASO and CAFO in Figure 1. Before taking the difference, we normalize the Δ solutions produced by CASO and CAFO to have the same L_2 norm because a constant scaling of elements of Δ does not change the visualization.

The empirical results are consistent with our theoretical results (Figure 1): the second-order interpretation results are similar to the first-order ones when the classification confidence probability is high. However, when the classification confidence probability is small, including the Hessian term can be useful in deep learning interpretation.

To observe the difference between CAFO and CASO interpretations in both regimes qualitatively, we compare them for an image when the confidence probability is high and for one where it is low in Figure 2. When the confidence probability is high, CAFO $\approx$ CASO and when this probability is low, CASO $\neq$ CAFO.

4. Understanding Impact of the group-feature

In this section, we study the impact of the group feature in deep learning interpretation. The group feature has been included as the sparsity constraint in optimization (2).

To obtain an unconstrained concave optimization for the CASO interpretation, we relaxed the sparsity (cardinality) constraint $\|\Delta\|_0 \leq k$ (often called an L_0 norm constraint) to a convex L_1 norm constraint. Such a $L_0 - L_1$ relaxation is a core component for popular learning methods such as compressive sensing (Candes & Tao, 2005; Donoho, 2006) or LASSO regression (Tibshirani, 1996). Using results from this literature, we show this relaxation is tight under certain conditions on the Hessian matrix $\mathbf{H}_\mathbf{x}$ (see Appendix Section E). In other words, the optimal Δ of optimization (5) is sparse with the proper choice of regularization parameters.

Note that the regularization term $-\lambda_1\|\Delta\|_1$ is a concave function for $\lambda_1 > 0$. Similarly due to Theorem 3, the CASO interpretation objective (5) is strongly concave.

One method for optimizing this objective is to apply the gradient descent method used in the second-order interpretation but with the addition of an L_1 regularization penalty. In our early experiments, we found that this procedure leads to poor convergence properties in practice. This is partially due to the non-smoothness of the L_1 regularization term.

To resolve this issue, we instead use the *proximal* gradient descent to compute a solution for CAFO and CASO when $\lambda_1 > 0$. Using the Nesterov momentum method and backtracking with proximal gradient descent gives a convergence rate of $\mathcal{O}(1/t^2)$ where t is the number of gradient updates (Appendix Section B). Proximal GD has been used in other deep learning problems including adversarial examples as well (e.g. (Chen et al., 2017)).

Below we explain how we use the proximal gradient descent to include the group features in deep learning interpretation. First, we write the objective function as the sum of a smooth and non-smooth function:

$$\tilde{\ell}(\Delta) = \underbrace{\nabla_\mathbf{x}\ell\left(f_{\theta^*}(\mathbf{x}), y\right)^t \Delta + \frac{1}{2}\Delta^t\mathbf{H}_\mathbf{x}\Delta - \lambda_2\|\Delta\|_2^2}_{\text{Smooth Part}} \underbrace{-\lambda_1\|\Delta\|_1}_{\text{Non-Smooth Part}}$$

Let $g(\Delta)$ be the smooth, $h(\Delta)$ be the non-smooth part:

$$g(\Delta) = \nabla_\mathbf{x}\ell\left(f_{\theta^*}(\mathbf{x}), y\right)^t \Delta + \frac{1}{2}\Delta^t\mathbf{H}_\mathbf{x}\Delta - \lambda_2\|\Delta\|_2^2$$
$$h(\Delta) = -\lambda_1\|\Delta\|_1$$
$$\tilde{\ell}(\Delta) = g(\Delta) + h(\Delta)$$

The gradient of the smooth objective is given by:

$$\nabla_\Delta g(\Delta) = \nabla_\mathbf{x}\ell\left(f_{\theta^*}(\mathbf{x}), y\right) + \mathbf{H}_\mathbf{x}\Delta - 2\lambda_2\Delta$$

The proximal mapping is given by:

$$\text{prox}_\alpha(x) = \arg\min_z \frac{1}{\alpha}\|x - z\|_2^2 + \lambda_1\|z\|_1$$
$$= \begin{cases} x + \lambda_1\alpha & x \leq -\lambda_1\alpha \\ 0 & -\lambda_1\alpha < x \leq \lambda_1\alpha \\ x - \lambda_1\alpha & \lambda_1\alpha < x \end{cases}$$

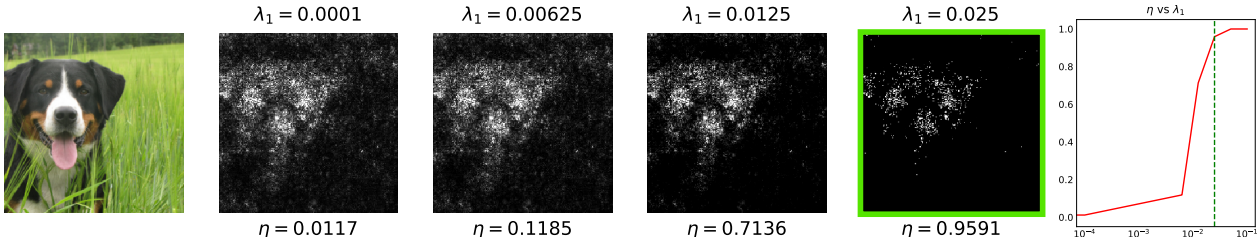
This formula can be understood intuitively as follows. If the magnitude of some elements of Δ is below a certain threshold ($\lambda_1\alpha$), proximal mapping sets those values to zero. This leads to values that are exactly zero in the saliency map. This can be viewed as removing noise by a certain thresholding procedure.

To optimize Δ, we use FISTA (Beck & Teboulle, 2009) with backtracking and the Nesterov momentum optimizer with a learning rate of 0.1 for 10 iterations and decay factor of 0.5. Δ is initialized to zero.

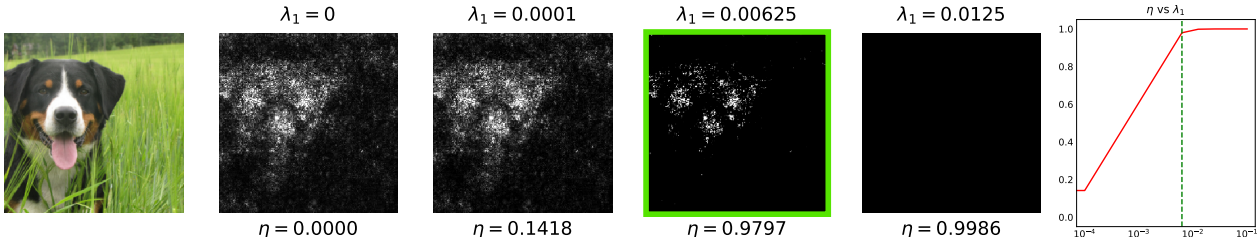
FISTA takes a step with learning rate α to reduce the smooth objective loss $g(\Delta)$, then applies a proximal mapping to the resulting Δ. Backtracking reduces the learning rate when the update results in higher loss.

4.1. Impact of group features in interpretation

In this section, our goal is to understand the impact of the group features in deep learning interpretation. In our experiments, we focus on the image classification problem because visual interpretations are intuitive and allow for comparison with prior work. We use a Resnet-50 (He et al., 2016) model on the ImageNet ILSVRC-2013 dataset (Russakovsky et al., 2015).



(a) Interpretation solutions for CAFO with different values of the regularization parameter λ_1



(b) Interpretation solutions for CASO with different values of the regularization parameter λ_1

Figure 3. Large λ_1 values lead to high sparsity ratios (η) of interpretation solutions. Our unsupervised method based on the sparsity ratio selects the interpretations marked with a green box in these examples.

To gain intuition for the effect of λ_1, we show a sweep over values in Figure 3. We observe that when λ_1 is set too high or too low, the interpretation breaks down as the importance values are relatively constant across the image (all high or all zero).

Different approaches to set the regularization parameter λ_1 have been explored in different problems. For example, in LASSO, one common approach is to use Least Angle Regression (Efron et al., 2004).

In the deep learning interpretation problem, we propose an unsupervised method based on the sparsity ratio of the interpretation solution to set a proper value for λ_1. We define η, the sparsity ratio, as the number of zero pixels divided by the total number of pixels. We tune λ_1 in an unsupervised fashion (since we do not know the ground truth interpretation) by increasing λ_1 until Δ reaches all zeros. We optimize with $\lambda_1 = [0, 10^{-5}, 10^{-4}, 10^{-3}, 6.25\times10^{-3}, 1.25\times10^{-2}, 2.5\times10^{-2}, 5\times10^{-2}]$. For interpretations with sparsity above a certain threshold (e.g. $\eta \geq 0.75$ in our examples), we choose the interpretation with the highest loss on the original model. In practice, we batch different values of λ_1 to find a reasonable parameter setting efficiently.

This method selects the interpretation marked with a green box in Figures 3a and 3b. We observe that adding group-feature terms makes the interpretation to be less noisy in these examples.

5. Qualitative Comparision of Deep learning Interpretation Methods

In this section, we briefly review prior approaches for the deep learning interpretation and compare their performance qualitatively. The proposed Hessian and group feature terms can be potentially included in these approaches as well.

Vanilla Gradient Simonyan et al. 2013 (Simonyan et al., 2014) propose to compute the gradient of the class score with respect to the input.

SmoothGrad Smilkov et al. 2017 (Smilkov et al., 2017) argues that the input gradient may fluctuate sharply in the region local to the test point. To address this, they average gradient-based importance values generated from many noisy versions of the input.

Integrated Gradients Sundararajan et al. 2017 (Sundararajan et al., 2017) define a *baseline*, which represents an input absent of information (e.g., a completely zero image). Feature importance is determined by accumulating gradient information along the path from the baseline to the original input: $(\mathbf{x}-\mathbf{x}') \times \int_{\alpha=0}^{1} \nabla_{\mathbf{x}} \ell\left(f_{\theta^*}(\mathbf{x}' + \alpha(\mathbf{x}-\mathbf{x}')), y\right) d\alpha$. The integral is approximated by a finite sum.

The idea of SmoothGrad (Smilkov et al., 2017) is to "smooth" the saliency map by averaging the importance values generated from many noisy versions of the input thereby smoothing the local fluctuations in the gradient. We use a similar idea to define smooth versions of CASO and CAFO. This yields the following interpretation objective.

Definition 4 (The Smooth CASO Interpretation) *For a*

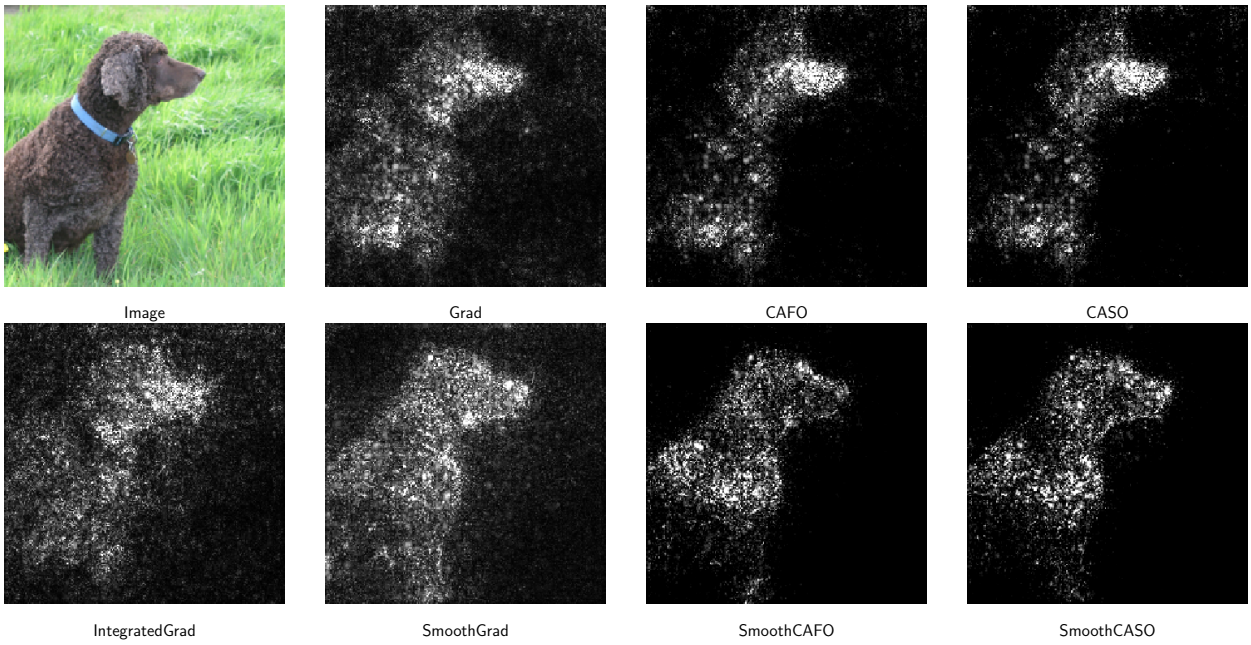


Figure 4. A qualitative comparison of some existing interpretation methods. More examples have been presented in Appendix Section G.

given sample $(\mathbf{x}, y)$*, we define the smooth context-aware second-order (the Smooth CASO) importance function* $\tilde{I}^{\lambda_1,\lambda_2}_{\theta^*}(\mathbf{x}, y)$ *as follows:*

$$\tilde{I}^{\lambda_1,\lambda_2}_{\theta^*}(\mathbf{x}, y) := \max_{\Delta} \; \frac{1}{n}\sum_{1}^{n}(\nabla_{\mathbf{z}}\ell\left(f_{\theta^*}(\mathbf{z}), y\right)^t \Delta + \frac{1}{2}\Delta^t \mathbf{H}_{\mathbf{z}}\Delta) - \lambda_1\|\Delta\|_1 - \lambda_2\|\Delta\|_2^2 \tag{8}$$

where $\mathbf{z} = \mathbf{x} + N(0, \sigma^2 I)$ *and* λ_1 *and* λ_2 *are defined similarly as before.*

In smoothed versions, we average over $n = 50$ number of noisy samples with σ set to 0.15. Smooth CAFO is defined similarly without Hessian term.

Since principled quantitative evaluations of saliency maps remain an open problem without properly annotated samples, we focus on some qualitative evaluations of different interpretation methods. Figure 4 shows a comparison between CAFO, CASO and other existing methods while more examples have been presented in Appendix Section G. We observe that including the group-feature in deep learning interpretation leads to a sparse saliency map, helping to eliminate the spurious noise and improving the quality of saliency maps.

6. Discussion

In this paper, we studied two aspects of the deep learning interpretation problem. First, by characterizing a closed-form formula for the Hessian matrix of a deep ReLU network, we showed that, if the confidence in the predicted class is high and the number of classes is large, first-order and second-order methods produce similar results. In the process, we also proved that the Hessian matrix is of rank one and its eigenvector is parallel to the gradient. These results can be insightful in other related problems such as adversarial examples. The extent of the Hessian impact for low confidence predictions and/or the case when the number of classes is small are among interesting directions for the future work. Second, we incorporated high-order feature dependencies in deep learning interpretation using a sparsity regularization term. This extension improves the deep learning interpretation significantly.

Although significant progresses have been made in tackling the deep learning interpretation problem, there remain some open problems as well. For example, since saleincy maps are high-dimensional vectors, they can be sensitive to noise and adversarial perturbations. Moreover, due to the lack of properly annotated datasets for the interpretation problem, the evaluation of interpretation methods are often qualitative and can be subjective. Resolving these issues are among interesting directions for the future work.

References

Adebayo, J., Gilmer, J., Goodfellow, I., and Kim, B. Local explanation methods for deep neural networks lack sensitivity to parameter values. In *ICLR Workshop*, 2018.

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., and Kim, B. Sanity Checks for Saliency Maps. *arXiv e-prints*, October 2018.

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., and Kim, B. Sanity checks for saliency maps. In *Proceedings of Advances in Neural Information Processing Systems*, 2018.

Alvarez-Melis, D. and Jaakkola, T. S. Towards Robust Interpretability with Self-Explaining Neural Networks. *arXiv e-prints*, June 2018.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., Samek, W., and Suárez, Ó. D. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. In *PloS one*, 2015.

Beck, A. and Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2009.

Bickel, P. J., Ritov, Y., Tsybakov, A. B., et al. Simultaneous analysis of lasso and dantzig selector. *The Annals of Statistics*, 2009.

Candes, E., Tao, T., et al. The dantzig selector: Statistical estimation when p is much larger than n. *The Annals of Statistics*, 2007.

Candes, E. J. and Tao, T. Decoding by linear programming. *IEEE transactions on information theory*, 2005.

Carlini, N. and Wagner, D. Towards Evaluating the Robustness of Neural Networks. *arXiv e-prints*, August 2016.

Chen, P.-Y., Sharma, Y., Zhang, H., Yi, J., and Hsieh, C.-J. Ead: elastic-net attacks to deep neural networks via adversarial examples. *arXiv preprint arXiv:1709.04114*, 2017.

Donoho, D. L. Compressed sensing. In *IEEE Transactions on Information Theory*, 2006.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. Least Angle Regression. *arXiv Mathematics e-prints*, June 2004.

Ghorbani, A., Abid, A., and Zou, J. Y. Interpretation of neural networks is fragile. *arXiv preprint arXiv: 1710.10547*, 2017.

Glorot, X., Bordes, A., and Bengio, Y. Deep sparse rectifier neural networks. In *Proceedings of Artificial Intelligence and Statistics*, 2011.

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. Maxout networks. In *Proceedings of the International Conference of Machine Learning*, 2013.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and Harnessing Adversarial Examples. *arXiv e-prints*, December 2014.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*, 2016.

Kindermans, P.-J., Schtt, K., Mller, K.-R., and Dhne, S. Investigating the influence of noise and distractors on the interpretation of neural networks. In *NIPS Workshop on Interpretable Machine Learning in Complex Systems*, 2016.

Kindermans, P.-J., Hooker, S., Adebayo, J., Alber, M., Schütt, K. T., Dähne, S., Erhan, D., and Kim, B. The (un)reliability of saliency methods. *arXiv preprint arXiv: 1711.00867*, 2017.

Li, J., Monroe, W., and Jurafsky, D. Understanding neural networks through representation erasure. *arXiv preprint arXiv: 1612.08220*, 2016.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. DeepFool: a simple and accurate method to fool deep neural networks. *arXiv e-prints*, November 2015.

Nie, W., Zhang, Y., and Patel, A. A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. *arXiv preprint arXiv:1805.07039*, 2018.

Parikh, N. and Boyd, S. Proximal algorithms. *Found. Trends Optim.*, 1(3):127–239, January 2014. ISSN 2167-3888. doi: 10.1561/2400000003. URL http://dx.doi.org/10.1561/2400000003.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. In *NIPS Autodiff Workshop: The Future of Gradient-based Machine Learning Software and Techniques*, 2017.

Pearlmutter, B. A. Fast exact multiplication by the hessian. In *Neural Computation*, 1994.

Raskutti, G., Wainwright, M. J., and Yu, B. Restricted eigenvalue properties for correlated gaussian designs. *Journal of Machine Learning Research*, 11(Aug):2241–2259, 2010.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 2015.

Shrikumar, A., Greenside, P., and Kundaje, A. Learning important features through propagating activation differences. In *Proceedings of the International Conference of Machine Learning*, 2017.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Proceedings of the International Conference on Learning Representations*, 2014.

Smilkov, D., Thorat, N., Kim, B., Viégas, F. B., and Wattenberg, M. SmoothGrad: removing noise by adding noise. *arXiv preprint arXiv: 1706.03825*, 2017.

Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In *Proceedings of the International Conference of Machine Learning*, 2017.

Tibshirani, R. Regression shrinkage and selection via the lasso. In *Journal of the Royal Statistical Society*, 1996.

Yao, Z., Gholami, A., Lei, Q., Keutzer, K., and Mahoney, M. W. Hessian-based analysis of large batch training and robustness to adversaries. *arXiv preprint arXiv:1802.08241*, 2018.

Appendix

A. Proofs

A.1. Proof of Proposition 1

In this section, we derive the closed-form formula for the Hessian of the loss function of a deep ReLU network. Since a ReLU network is piecewise linear, it is locally linear around an input $\mathbf{x}$. Thus the logits can be represented as:

$$f_\theta(\mathbf{x}) = \mathbf{W}^T\mathbf{x} + \mathbf{b},$$

where $\mathbf{x}$ is the input of dimension d, $f_\theta(\mathbf{x})$ are the logits, $\mathbf{W}$ are the weights, and $\mathbf{b}$ are the biases of the linear function. In this proof, we use $\hat{\mathbf{y}}$ to denote the logits, $\mathbf{p}$ to denote the class probabilities, $\mathbf{y}$ to denote the label vector and c to denote the number of classes. Each column $\mathbf{W}_i$ of $\mathbf{W}$ is the gradient of logit $\hat{\mathbf{y}}_i$ with respect to flattened input $\mathbf{x}$ and can be easily handled in auto-grad software such as PyTorch (Paszke et al., 2017).

Thus

$$\frac{\partial \hat{\mathbf{y}}_i}{\partial \mathbf{x}} = \mathbf{W}_i \tag{9}$$

$$\mathbf{p} = \text{softmax}(\hat{\mathbf{y}})$$

$$\ell(\mathbf{p}, \mathbf{y}) = -\sum_{i=1}^{c} \mathbf{y}_i \log(\mathbf{p}_i).$$

$$\nabla_{\hat{\mathbf{y}}} \ell(\mathbf{p}, \mathbf{y}) = \mathbf{p} - \mathbf{y}$$

$$\implies \frac{\partial \ell(\mathbf{p}, \mathbf{y})}{\partial \hat{\mathbf{y}}_i} = \mathbf{p}_i - \mathbf{y}_i \tag{10}$$

$$\nabla_{\mathbf{x}} \ell(\mathbf{p}, \mathbf{y}) = \sum_{i=1}^{c} \frac{\partial \hat{\mathbf{y}}_i}{\partial \mathbf{x}} \times \frac{\partial \ell(\mathbf{p}, \mathbf{y})}{\partial \hat{\mathbf{y}}_i}$$

Using (9) and (10),

$$\nabla_{\mathbf{x}} \ell(\mathbf{p}, \mathbf{y}) = \sum_{i=1}^{c} \mathbf{W}_i(\mathbf{p}_i - \mathbf{y}_i)$$

$$\implies \nabla_{\mathbf{x}} \ell(\mathbf{p}, \mathbf{y}) = \mathbf{W}(\mathbf{p} - \mathbf{y})$$

Therefore, we have:

$$\mathbf{H}_{\mathbf{x}} = \nabla_{\mathbf{x}}(\nabla_{\mathbf{x}} \ell(\mathbf{p}, \mathbf{y})) = \nabla_{\mathbf{x}}(\sum_{i=1}^{c} \mathbf{W}_i(\mathbf{p}_i - \mathbf{y}_i))$$

$$\mathbf{H}_{\mathbf{x}} = \sum_{i=1}^{c} \mathbf{W}_i(\nabla_{\mathbf{x}}(\mathbf{p}_i - \mathbf{y}_i))^T$$

$$\mathbf{H}_{\mathbf{x}} = \sum_{i=1}^{c} \mathbf{W}_i(\nabla_{\mathbf{x}} \mathbf{p}_i)^T \tag{11}$$

Deriving $\nabla_{\mathbf{x}} \mathbf{p}_i$:

$$\nabla_{\mathbf{x}} \mathbf{p}_i = \sum_{j=1}^{c} \frac{\partial \hat{\mathbf{y}}_j}{\partial \mathbf{x}} \times \frac{\partial \mathbf{p}_i}{\partial \hat{\mathbf{y}}_j}$$

$$\implies \nabla_{\mathbf{x}} \mathbf{p}_i = \sum_{j=1}^{c} \left(\mathbf{W}_j \times \frac{\partial \mathbf{p}_i}{\partial \hat{\mathbf{y}}_j} \right) \quad \text{(Using (9))} \tag{12}$$

$$\frac{\partial \mathbf{p}_i}{\partial \hat{\mathbf{y}}_j} = \begin{cases} \mathbf{p}_i - \mathbf{p}_i^2 & i = j \\ -\mathbf{p}_i \mathbf{p}_j & i \neq j \end{cases}$$

$$\implies \nabla_{\hat{\mathbf{y}}} \mathbf{p} = \text{diag}(\mathbf{p}) - \mathbf{p}\mathbf{p}^T \tag{13}$$

$$\mathbf{H}_{\mathbf{x}} = \sum_{i=1}^{c} \mathbf{W}_i \left(\sum_{j=1}^{c} \mathbf{W}_j \times \frac{\partial \mathbf{p}_i}{\partial \hat{\mathbf{y}}_j} \right)^T \quad \text{(Substituting (12) in (11))}$$

$$\mathbf{H}_{\mathbf{x}} = \sum_{i=1}^{c} \sum_{j=1}^{c} \mathbf{W}_i \frac{\partial \mathbf{p}_i}{\partial \hat{\mathbf{y}}_j} \mathbf{W}_j^T$$

$$\implies \mathbf{H}_{\mathbf{x}} = \mathbf{W}(\text{diag}(\mathbf{p}) - \mathbf{p}\mathbf{p}^T)\mathbf{W}^T \quad \text{(Using (13))}$$

Thus we have,

$$\nabla_{\mathbf{x}} \ell(\mathbf{p}, \mathbf{y}) = \mathbf{g}_{\mathbf{x}} = \mathbf{W}(\mathbf{p} - \mathbf{y}) \tag{14}$$

$$\mathbf{H}_{\mathbf{x}} = \mathbf{W}\mathbf{A}\mathbf{W}^T \tag{15}$$

where

$$\mathbf{A} \coloneqq \text{diag}(\mathbf{p}) - \mathbf{p}\mathbf{p}^T. \tag{16}$$

This completes the proof.

A.2. Proof of Theorem 2

To simplify notation, define $\mathbf{A}$ as in (16). For any arbitrary row of the matrix $\mathbf{A}_i$, we have

$$\begin{aligned}
&\sum_{\mathrm{j}\neq\mathrm{i}} |\mathbf{A}_{ij}| = (\sum_{\mathrm{j}\neq\mathrm{i}} |-\mathbf{p}_i\mathbf{p}_j|) \\
\implies &\sum_{\mathrm{j}\neq\mathrm{i}} |\mathbf{A}_{ij}| = \mathbf{p}_i \sum_{\mathrm{j}\neq\mathrm{i}} \mathbf{p}_j \\
\implies &\sum_{\mathrm{j}\neq\mathrm{i}} |\mathbf{A}_{ij}| = \mathbf{p}_i(1-\mathbf{p}_i) \\
&|\mathbf{A}_{ii}| = \mathbf{p}_i(1-\mathbf{p}_i)
\end{aligned}$$

Because $|\mathbf{A}_{ii}| >= \sum_{\mathrm{j}\neq\mathrm{i}} |\mathbf{A}_{ij}|$, by the Gershgorin Circle theorem, we have that all eigenvalues of $\mathbf{A}$ are positive and $\mathbf{A}$ is a positive semidefinite matrix. Since $\mathbf{A}$ is psd, we can write $\mathbf{A} = \mathbf{L}\mathbf{L}^T$. Using (15):

$$\mathbf{H}_\mathbf{x} = \mathbf{W}\mathbf{A}\mathbf{W}^T = \mathbf{W}\mathbf{L}\mathbf{L}^T\mathbf{W}^T = \mathbf{W}\mathbf{L}(\mathbf{W}\mathbf{L})^T$$

Hence $\mathbf{H}_\mathbf{x}$ is a positive semidefinite matrix as well.

A.3. Proof of Theorem 3

The second-order interpretation objective function is given by,

$$\begin{aligned}
\tilde{\ell}(\Delta) &= \nabla_\mathbf{x}\ell\left(f_{\theta^*}(\mathbf{x}), y\right)^t \Delta + \frac{1}{2}\Delta^t\mathbf{H}_\mathbf{x}\Delta - \lambda_2\|\Delta\|_2^2 \\
\tilde{\ell}(\Delta) &= \nabla_\mathbf{x}\ell\left(f_{\theta^*}(\mathbf{x}), y\right)^t \Delta + \frac{1}{2}\Delta^t(\mathbf{H}_\mathbf{x} - 2\lambda_2\mathbf{I})\Delta
\end{aligned}$$

where $\Delta \coloneqq \tilde{\mathbf{x}} - \mathbf{x}$ (y is fixed). Therefore if $\lambda_2 > L/2$, $\mathbf{H}_\mathbf{x} - 2\lambda_2\mathbf{I}$ is negative definite and $\tilde{\ell}(\Delta)$ is strongly concave.

A.4. Proof of Theorem 4

Let the class probabilities be denoted by $\mathbf{p}$, the number of classes by c and the label vector by $\mathbf{y}$. We again use $\mathbf{g}_\mathbf{x}$ and $\mathbf{H}_\mathbf{x}$ as defined in (14) and (15) respectively. Without loss of generality, assume that the first class is the one with maximum probability.

$$\text{Hence, } \mathbf{y} = [1, 0, 0, ..., 0]^T \tag{17}$$

We assume all other classes have small probability,

$$\begin{aligned}
&\mathbf{p}_i = \epsilon \approx 0 \ \forall\ \mathrm{i} \in [2, \mathrm{c}] \\
&\text{Since } \sum_{i=1}^{c} \mathbf{p}_i = 1, \implies \mathbf{p}_1 = 1 - (c-1)\epsilon, \\
&\implies \mathbf{p} = \left[1-(c-1)\epsilon,\ \epsilon, \ldots,\ \epsilon\right]^T,
\end{aligned} \tag{18}$$

We define:

$$\begin{aligned}
&\mathbf{A} = \text{diag}(\mathbf{p}) - \mathbf{p}\mathbf{p}^T \text{ where,} \\
&\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1c} \\ a_{21} & a_{22} & \ldots & a_{2c} \\ \vdots & \vdots & \ddots & \vdots \\ a_{c1} & a_{c2} & \ldots & a_{cc} \end{bmatrix} \\
&a_{11} = 1 - (c-1)\epsilon - (1-(c-1)\epsilon)^2 \\
&a_{1i} = a_{i1} = -(1-(c-1)\epsilon)\epsilon\ \forall i \in [2, \mathrm{c}] \\
&a_{ii} = \epsilon - \epsilon^2\ \forall i \in [2, \mathrm{c}] \\
&a_{ij} = -\epsilon^2\ \forall i, j \in [2, \mathrm{c}],\ \mathrm{i} \neq \mathrm{j}
\end{aligned}$$

Ignoring ϵ^2 terms:

$$\begin{aligned}
&a_{11} = (c-1)\epsilon \\
&a_{1i} = a_{i1} = -\epsilon\ \forall i \in [2, \mathrm{c}] \\
&a_{ii} = \epsilon\ \forall i \in [2, \mathrm{c}] \\
&a_{ij} = 0\ \forall i, j \in [2, \mathrm{c}],\ \mathrm{i} \neq \mathrm{j}
\end{aligned}$$

Let λ be an eigenvalue of $\mathbf{A}$ and $\mathbf{v}$ be an eigenvector of $\mathbf{A}$, then $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$.
Let $v_1, v_2, \ldots, v_n$ be the individual components of the eigenvector. The equation $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$ can be rewritten in terms of its individual components as follows:

$$(c-1)\epsilon v_1 - \epsilon\sum_{i=2}^{c} v_i = \lambda v_1 \tag{19}$$

$$\begin{aligned}
&-\epsilon v_1 + \epsilon v_i = \lambda v_i\ \forall i \in [2, c] \\
&\implies v_i = \frac{\epsilon}{\epsilon - \lambda} v_1\ \forall i \in [2, c], \text{ for } \lambda \neq \epsilon
\end{aligned} \tag{20}$$

$$\implies \text{or } v_1 = 0, \text{ for } \lambda = \epsilon \tag{21}$$

We first consider the case $\lambda \neq \epsilon$ (20).
Substituting v_i in (19):

$$\begin{aligned}
(c-1)\epsilon v_1 - \epsilon\sum_{i=2}^{c} v_i &= (c-1)\epsilon v_1 - \frac{\epsilon^2}{\epsilon-\lambda}\sum_{i=2}^{c} v_1 \\
&= (c-1)\epsilon v_1 - \frac{\epsilon^2}{\epsilon-\lambda}(c-1)v_1 \\
&= (c-1)\epsilon v_1 - (c-1)\epsilon v_1\frac{\epsilon}{\epsilon-\lambda} \\
&= \lambda v_1
\end{aligned}$$

$$(c-1)\epsilon v_1\left[1-\frac{\epsilon}{\epsilon-\lambda}\right]=\lambda v_1$$

$$(c-1)\epsilon v_1\left[-\frac{\lambda}{\epsilon-\lambda}\right]=\lambda v_1$$

$$(c-1)\epsilon v_1(-\lambda)=\lambda v_1(\epsilon-\lambda)$$

$$\implies \lambda v_1(c\epsilon-\lambda)=0$$

$$\implies \lambda=0 \text{ or } v_1=0 \text{ or } \lambda=c\epsilon$$

But, $v_1=0 \implies v_i=\frac{\epsilon}{\epsilon-\lambda}v_1=0\ \forall\ i\in[2,c]$

$$\implies \mathbf{v}=0$$

Since $\mathbf{v}$ is an eigenvector, it cannot be zero,

$$\implies \lambda=0 \text{ or } \lambda=c\epsilon.$$

Let $\mathbf{u}_1$ be the corresponding eigenvector for $\lambda=c\epsilon$.

By substituting $\lambda=c\epsilon$ in (20)

$$\mathbf{u}_1^T \propto [1-c,1,...,1] \tag{22}$$

Dividing by the normalization constant, (23)

$$\mathbf{u}_1^T=\frac{1}{\sqrt{c(c-1)}}[1-c,1,...,1]$$

Now we consider the case $\lambda=\epsilon$ (21),
Substituting $v_1=0, \lambda=\epsilon$ in (19):

The space of eigenvectors for $\lambda=\epsilon$ is an $c-2$ dimensional subspace with $v_1=0, \sum_{i=2}^{c} v_i=0$.

Let $\mathbf{u}_i$ be the eigenvectors with $\lambda=\epsilon \quad \forall$ i $\in$ [2, c-1]

Let $\mathbf{u}_c$ be the eigenvector with $\lambda=0$.

Writing $\mathbf{A}$ in terms of its eigenvalues and eigenvectors,

$$\mathbf{A}=c\epsilon\mathbf{u}_1\mathbf{u}_1^T+\epsilon\sum_{i=2}^{c-1}\mathbf{u}_i\mathbf{u}_i^T$$

Let $\mathbf{A}_1=c\epsilon\mathbf{u}_1\mathbf{u}_1^T$

Let $\mathbf{A}_2=\epsilon\sum_{i=2}^{c-1}\mathbf{u}_i\mathbf{u}_i^T$

$\|\mathbf{A}_1\|_F=c\epsilon, \|\mathbf{A}_2\|_F=\epsilon\sqrt{c-2}$

Hence as, $c\to\infty$,

$\mathbf{A}=\mathbf{A}_1+\mathbf{A}_2\approx\mathbf{A}_1$

Using (15),

$\mathbf{H}_\mathbf{x}=\mathbf{W}\mathbf{A}\mathbf{W}^T\approx\mathbf{W}\mathbf{A}_1\mathbf{W}^T$

Substituting $\mathbf{A}_1=c\epsilon\mathbf{u}_1\mathbf{u}_1^T$,

$$\mathbf{H}_\mathbf{x}\approx c\epsilon\mathbf{W}\mathbf{u}_1\mathbf{u}_1^T\mathbf{W}^T \tag{24}$$

Using (14),

$\mathbf{g}_\mathbf{x}=\nabla_\mathbf{x}\ell(\mathbf{p},\mathbf{y})=\mathbf{W}(\mathbf{p}-\mathbf{y})$

Let $\mathbf{W}_i$ denote the ith row of $\mathbf{W}$,

Using (17) and (18),

$$\mathbf{g}_\mathbf{x}=\mathbf{W}_1(1-c)\epsilon+\sum_{i=2}^{c}\mathbf{W}_i\epsilon$$

$$\mathbf{g}_\mathbf{x}=\epsilon(\mathbf{W}_1(1-c)+\sum_{i=2}^{c}\mathbf{W}_i)$$

Using (22),

$$\mathbf{g}_\mathbf{x}=\epsilon\sqrt{c(c-1)}\mathbf{W}\mathbf{u}_1$$

$$\implies \mathbf{W}\mathbf{u}_1=\frac{\mathbf{g}_\mathbf{x}}{\epsilon\sqrt{c(c-1)}} \tag{25}$$

Using (24),

$$\mathbf{H}_\mathbf{x}\approx c\epsilon\mathbf{W}\mathbf{u}_1\mathbf{u}_1^T\mathbf{W}^T=c\epsilon\mathbf{W}\mathbf{u}_1(\mathbf{W}\mathbf{u}_1)^T$$

Using (25),

$$\mathbf{H}_\mathbf{x}\approx c\epsilon\frac{\mathbf{g}_\mathbf{x}}{\epsilon\sqrt{c(c-1)}}\frac{\mathbf{g}_\mathbf{x}^T}{\epsilon\sqrt{c(c-1)}}$$

$$\mathbf{H}_\mathbf{x}\approx c\epsilon\frac{\mathbf{g}_\mathbf{x}\mathbf{g}_\mathbf{x}^T}{\epsilon^2c(c-1)}=\frac{\mathbf{g}_\mathbf{x}\mathbf{g}_\mathbf{x}^T}{\epsilon(c-1)}$$

$$\implies \mathbf{H}_\mathbf{x}\approx\frac{\mathbf{g}_\mathbf{x}\mathbf{g}_\mathbf{x}^T}{\epsilon(c-1)} \tag{26}$$

Thus, the Hessian is approximately rank one and the gradient is parallel to the Hessian's only eigenvector.

A.5. Proof of Theorem 5

We use $\mathbf{g}_\mathbf{x}=\nabla_\mathbf{x}\ell(\mathbf{p},\mathbf{y})=\mathbf{W}(\mathbf{p}-\mathbf{y})$ for simplicity (14).
When $\lambda_1=0$ in the CASO and CAFO objectives:

The CASO objective becomes:

$$\max_{\Delta}\ (\mathbf{g}_\mathbf{x}^t\Delta+\frac{1}{2}\Delta^t\mathbf{H}_\mathbf{x}\Delta-\lambda_2\|\Delta\|_2^2)$$

Taking the derivative with respect to Δ and solving :

$$\Delta^*_{CASO}=(2\lambda_2\mathbf{I}-\mathbf{H}_\mathbf{x})^{-1}\mathbf{g}_\mathbf{x}$$

Similarly, for the CAFO objective we get :

$$\Delta^*_{CAFO}=\frac{1}{2\lambda_2}\mathbf{g}_\mathbf{x}$$

Using (26),

$$\mathbf{H}_\mathbf{x}\approx\frac{\mathbf{g}_\mathbf{x}\mathbf{g}_\mathbf{x}^T}{\epsilon(c-1)}=\frac{\|\mathbf{g}_\mathbf{x}\|^2}{\epsilon(c-1)}\frac{\mathbf{g}_\mathbf{x}\mathbf{g}_\mathbf{x}^T}{\|\mathbf{g}_\mathbf{x}\|^2}$$

Define $\mu=\frac{\|\mathbf{g}_\mathbf{x}\|^2}{\epsilon(c-1)}$. Thus μ is the eigenvalue of $\mathbf{H}_\mathbf{x}$ for the eigenvector $\frac{\mathbf{g}_\mathbf{x}}{\|\mathbf{g}_\mathbf{x}\|}$.

Consider the matrix $\mathbf{B}=(2\lambda_2\mathbf{I}-\mathbf{H}_\mathbf{x})$:

Let $\mathbf{z}_1, \ldots, \mathbf{z}_d$ be the eigenvectors of $\mathbf{B}$,

where $\mathbf{z}_1 = \dfrac{\mathbf{g_x}}{\|\mathbf{g_x}\|}$

Eigenvalue for $\mathbf{z}_1 = 2\lambda_2 - \mu$

Eigenvalue for $\mathbf{z}_i = 2\lambda_2 \quad \forall i \in [2, d]$

$$\mathbf{B} = (2\lambda_2 - \mu)\mathbf{z}_1\mathbf{z}_1^T + 2\lambda_2 \sum_{i=2}^{i=d} \mathbf{z}_i\mathbf{z}_i^T$$

$$\mathbf{B}^{-1} = \frac{1}{(2\lambda_2 - \mu)}\mathbf{z}_1\mathbf{z}_1^T + \frac{1}{2\lambda_2}\sum_{i=2}^{i=d} \mathbf{z}_i\mathbf{z}_i^T$$

$$\mathbf{B}^{-1} = \frac{1}{(2\lambda_2 - \mu)}\frac{\mathbf{g_x}\mathbf{g_x}^T}{\|\mathbf{g_x}\|^2} + \frac{1}{2\lambda_2}\sum_{i=2}^{i=d} \mathbf{z}_i\mathbf{z}_i^T$$

$$\Delta^*_{CASO} = \mathbf{B}^{-1}\mathbf{g_x}$$

$$\Delta^*_{CASO} = \left[\frac{1}{(2\lambda_2 - \mu)}\frac{\mathbf{g_x}\mathbf{g_x}^T}{\|\mathbf{g_x}\|^2} + \frac{1}{2\lambda_2}\sum_{i=2}^{i=d} \mathbf{z}_i\mathbf{z}_i^T\right]\mathbf{g_x}$$

Since each $\mathbf{z}_i$ is orthogonal to $\mathbf{g_x}$

$$\implies \Delta^*_{CASO} = \frac{\mathbf{g_x}}{(2\lambda_2 - \mu)} = \frac{2\lambda_2\Delta^*_{CAFO}}{(2\lambda_2 - \mu)}$$

Hence $\Delta^*_{CASO} \parallel \Delta^*_{CAFO}$ and since scaling does not affect the visualization, the two interpretations are equivalent.

B. Convergence of Gradient Descent to Solve CASO

A consequence of Theorem 3 is that gradient descent converges to the global optimizer of the second-order interpretation objective objective with a convergence rate of $\mathcal{O}(1/t^2)$. More precisely, we have:

Corollary 1 *Let $\tilde{\ell}(\Delta)$ be the objective function of the second-order interpretation objective defined in Section 2 (Definition 3). Let $\Delta^{(t)}$ be the value of Δ in the t^{th} step with a learning rate $\alpha \le \lambda_2 - L/2$. We have*

$$\tilde{\ell}(\Delta^{(t)}) - \tilde{\ell}(\Delta^*) \le \frac{2\|\Delta^{(0)} - \Delta^*\|_2^2}{\alpha(t+1)^2}.$$

C. Efficient Computation of the Hessian Matrix Using the Cholesky decomposition

By Theorem 2, the Cholesky decomposition of $\mathbf{A}$ (defined in (16)) exists. Let $\mathbf{L}$ be the Cholesky decomposition of $\mathbf{A}$. Thus, we have

$$\mathbf{A} = \mathbf{L}\mathbf{L}^T$$
$$\mathbf{H_x} = \mathbf{W}\mathbf{L}\mathbf{L}^T\mathbf{W}^T$$

Let $\mathbf{B} := \mathbf{W}\mathbf{L}$. Thus, $\mathbf{H_x}$ can be re-written as $\mathbf{H_x} = \mathbf{B}\mathbf{B}^T$.

Let the SVD of $\mathbf{B}$ be as the following:

$$\mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

Thus, we can write:

$$\mathbf{H_x} = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^T$$

Define $\mathbf{C} = \mathbf{B}^T\mathbf{B} = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}$. Note that $\mathbf{\Sigma}^2$, the eigenvalues of $\mathbf{C}$ and $\mathbf{H_x}$ are the same. For a dataset such as ImageNet, the input has dimension d = 224×224×3 and c = 1000. Decomposing C (size 1000×1000) into its eigenvalues $\mathbf{\Sigma}$ and eigenvectors $\mathbf{V}$ is computationally efficient. Thus, from $\mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, we can compute the eigenvectors $\mathbf{U}$ of $\mathbf{H_x}$.

D. Saliency Visualization Methods

Normalizing Feature Importance Values After assigning importance values to each input feature, the values must be normalized for visualization in a saliency map. For fair comparison across all methods, we use the non-diverging normalization method for SmoothGrad (Smilkov et al., 2017). This normalization method first takes the absolute value of the importance scores and sums across the three color channels of the image. Next, the largest importance values are capped to the value of 99^{th} percentile. Finally, the importance values are divided and clipped to enforce the range $[0, 1]$.[3]

Domain-Specific Post-Processing Gradient $\odot$ Input (Shrikumar et al., 2017) multiplies the importance values by the raw feature values. In image tasks where the baseline is zero, Integrated Gradients (Sundararajan et al., 2017) does the same. This heuristic can visually sharpen the saliency map and has some theoretical justification: it is equivalent to the original Layerwise Relevance Propagation Technique (Bach et al., 2015) modulo a scaling factor (Kindermans et al., 2016; Shrikumar et al., 2017). Additionally, if the model is linear, $y = W\mathbf{x}$, mutliplying the gradient by the input is equivalent to a feature's true contribution to the final class score.

However, multiplying by the input can introduce visual artifacts not present in the importance values (Smilkov et al., 2017). We argue against multiplying by the input: it artificially enhances the visualization and only yields benefits in the image domain. Adebayo et al. 2018 (Adebayo et al., 2018) argue similar and show cases when the input term can dominate the interpretation. Moreover, multiplication by the input removes the input invariance of the interpretation regardless of the invariances of the underlying model (**?**). We observed numerous failures in existing interpretation methods when input multiplication is removed.

[3] https://github.com/PAIR-code/saliency/blob/master/saliency/visualization.py

E. Tightness of the $L_0 - L_1$ Relaxation

We assume the condition of Theorem 3 holds, thus, the CASO optimization is a concave maximization (equivalently a convex minimization) problem.

Note the CASO optimization with the cardinality constraint can be re-written as follows:

$$\min_{\Delta} \quad \|\mathbf{y} - \mathbf{A}\Delta\|^2, \qquad \|\Delta\|_0 \leq k, \tag{27}$$

where

$$\mathbf{A} := \left(\lambda_2 \mathbf{I} - \frac{1}{2}\mathbf{H}_{\mathbf{x}}\right)^{1/2} \tag{28}$$

$$\mathbf{y} := \frac{1}{2}\mathbf{A}^{-1}\nabla_{\mathbf{x}}\ell\left(f_{\theta^*}(\mathbf{x}), y\right). \tag{29}$$

$(.)^{1/2}$ indicates the square root of a positive definite matrix. Equation (28) highlights the condition for tuning the parameter λ_2: it needs to be sufficiently large to allow inversion of $\mathbf{A}$ but sufficiently small to not overpower the Hessian term. Note, we are now minimizing Δ for consistency with the compressive sensing literature. To explain the conditions under which the $L_0 - L_1$ relaxation is tight, we define the following notation. For a given subset $S \subset \{1, 2, ..., d\}$ and constant $\alpha \geq 1$, we define the following cone:

$$\mathcal{C}(S; \alpha) := \{\Delta \in \mathbb{R}^d : \|\Delta_{S^c}\|_1 \leq \|\Delta_S\|_1\}, \tag{30}$$

where S^c is the complement of S. We say that the matrix $\mathbf{A}$ satisfies the restricted eigenvalue (RE) (Bickel et al., 2009; Raskutti et al., 2010) condition over S with parameters $(\alpha, \gamma) \in [1, \infty) \times (0, \infty)$ if

$$\frac{1}{d}\|\mathbf{A}\Delta\|_2^2 \geq \gamma^2 \|\Delta\|_2^2 \quad \forall \Delta \in \mathcal{C}(S; \alpha). \tag{31}$$

If this condition is satisfied for all subsets of S where $|S| = k$, we say that $\mathbf{A}$ satisfies the RE condition of order k with parameters (α, γ). If $\mathbf{A}$ satisfies the RE condition with $\alpha \geq 3$ and $\gamma > 0$, then the $L_0 - L_1$ relaxation of optimization (27) is tight (Bickel et al., 2009). In other words, if Δ^* is the solution of optimization (27), it is also the solution of optimization

$$\min_{\Delta} \quad \|\mathbf{y} - \mathbf{A}\Delta\|^2, \qquad \|\Delta\|_1 \leq \|\Delta^*\|_1. \tag{32}$$

The Lagrange relaxation of this optimization leads to the CASO interpretation objective. We note that the RE condition is less severe than other optimality conditions such as the restricted isometry property (Candes et al., 2007). Although it is difficult to verify that the RE condition holds for the Hessian matrix of a deep neural network, empirical experiments are consistent with the theoretical results: the resulting Δ of the CASO interpretation objective is sparse for proper choices of the regularization parameters as shown below.

F. Additional Details on Experiments

F.1. Details on Experiments reported in Figure 1

Current autograd software does not support fast eigenvalue decomposition of a matrix in a batched setting making the exact hessian implementation inefficient when we need to compute the interpretations for a large number of samples. Hence for the purposes of this experiment, we use Proximal gradient descent to compute the interpretations Δ_{CASO} and Δ_{CAFO}, even though the parameter λ_1 is set to zero. Details for the other hyperparameters are given in Table 1.

Table 1. Hyper-parameter details for Figure 1

Parameter	Config
λ_1	0
λ_2 threshold	20
Optimizer	Proximal Gradient Descent
Network architecture	Resnet-50
Batch size	32
Power method iterations	10
Gradient descent iterations	10
Backtracking decay factor	0.5
Initialization	Zero

F.2. Details on Experiments reported in Figure 3

Details of the hyperparameters are in Table 2

Table 2. Hyper-parameter details for Figure 3

Parameter	Config
λ_1 values	0, 10^{-5}, 10^{-4}, 10^{-3}, 6.25×10^{-3}, 1.25×10^{-2}, 2.5×10^{-2}, 5×10^{-2}
λ_2 threshold	20
Optimizer	Proximal Gradient Descent
Network architecture	Resnet-50
Batch size	32
Power method iterations	10
Gradient descent iterations	10
Backtracking decay factor	0.5
Initialization	Zero

F.3. Details on Experiments reported in Figure 4

Details of the hyperparameters are in Table 3

Table 3. Hyper-parameter details for Figure 4

Parameter	Config
λ_1 values	0, 10^{-5}, 10^{-4}, 10^{-3}, 6.25×10^{-3}, 1.25×10^{-2}, 2.5×10^{-2}, 5×10^{-2}
λ_2 threshold	20
Optimizer	Proximal Gradient Descent
Network architecture	Resnet-50
Batch size	32
Power method iterations	10
Gradient descent iterations	10
Backtracking decay factor	0.5
Number of samples	32
Stddev of Random samples	0.15
Initialization	Zero

G. Comparison with existing methods

Additional comparison is given in figures 5, 6, 7, 8, 9, 10.

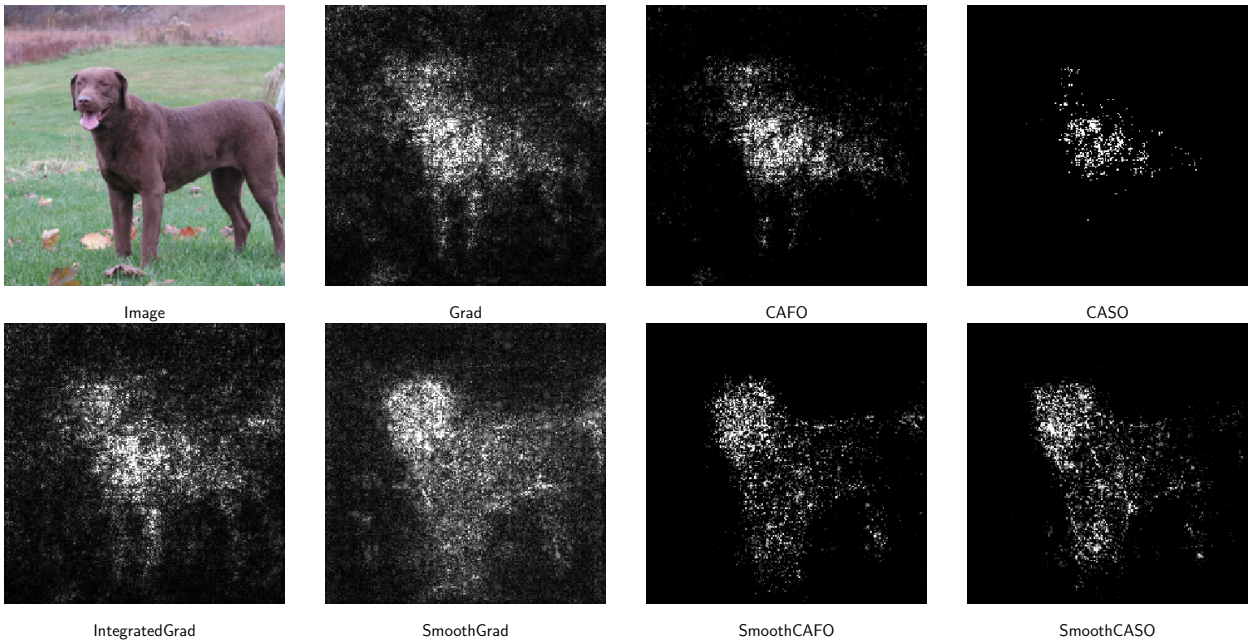


Figure 5.

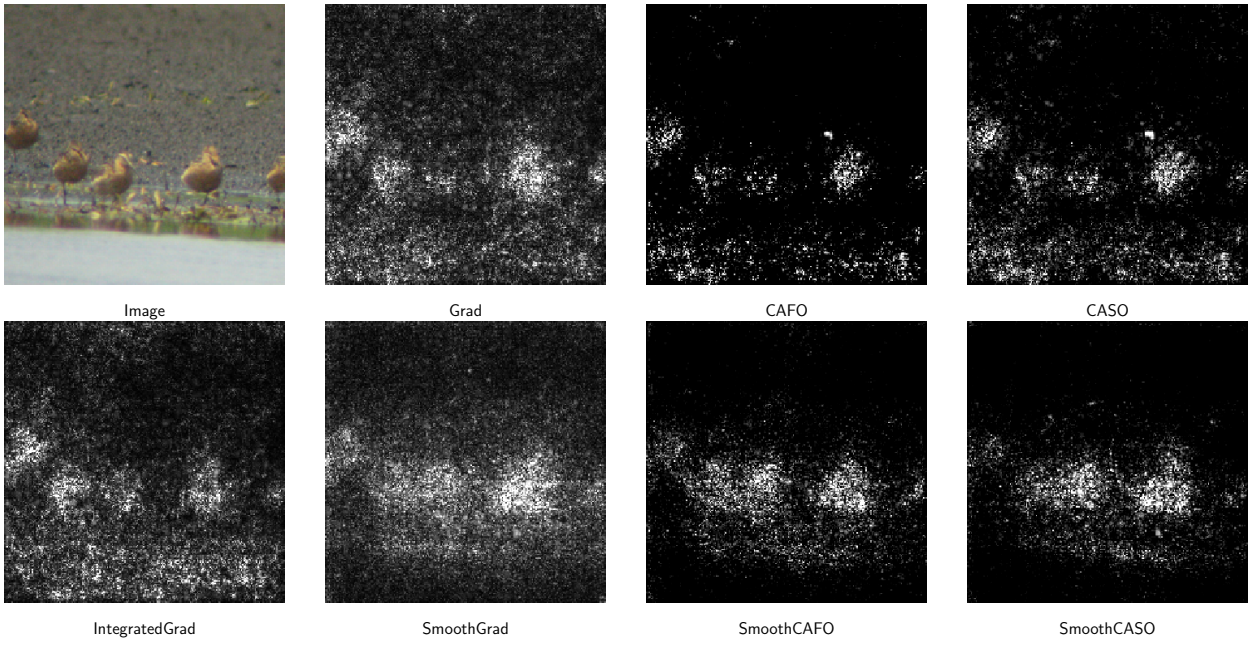


Figure 6.

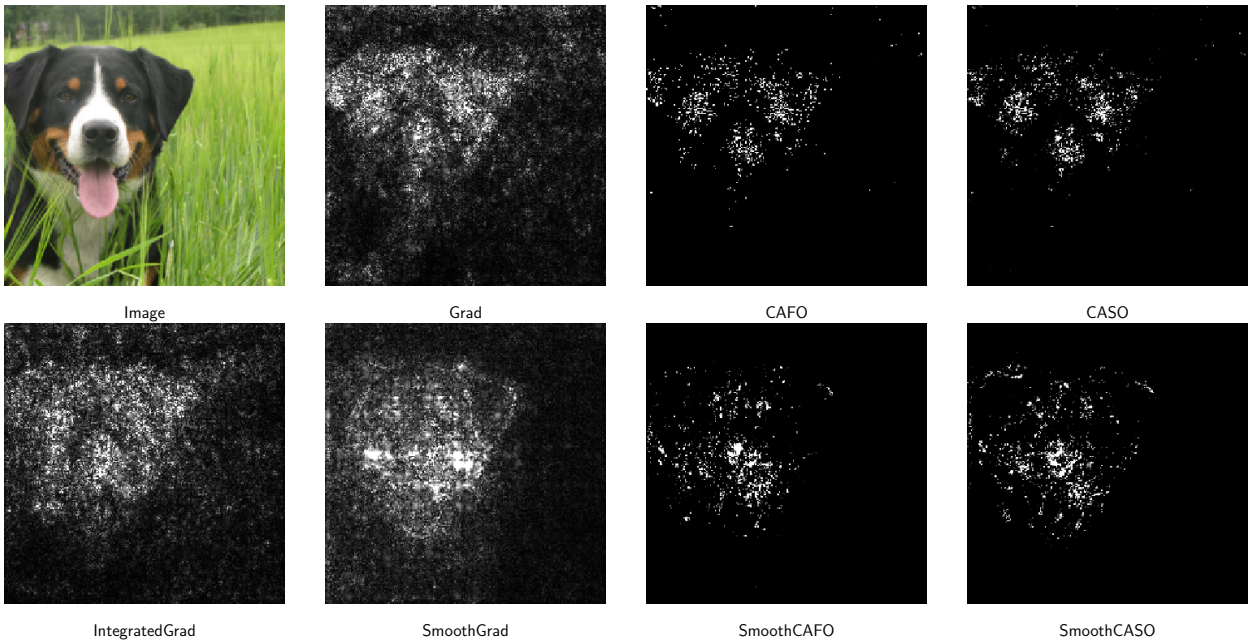


Figure 7.

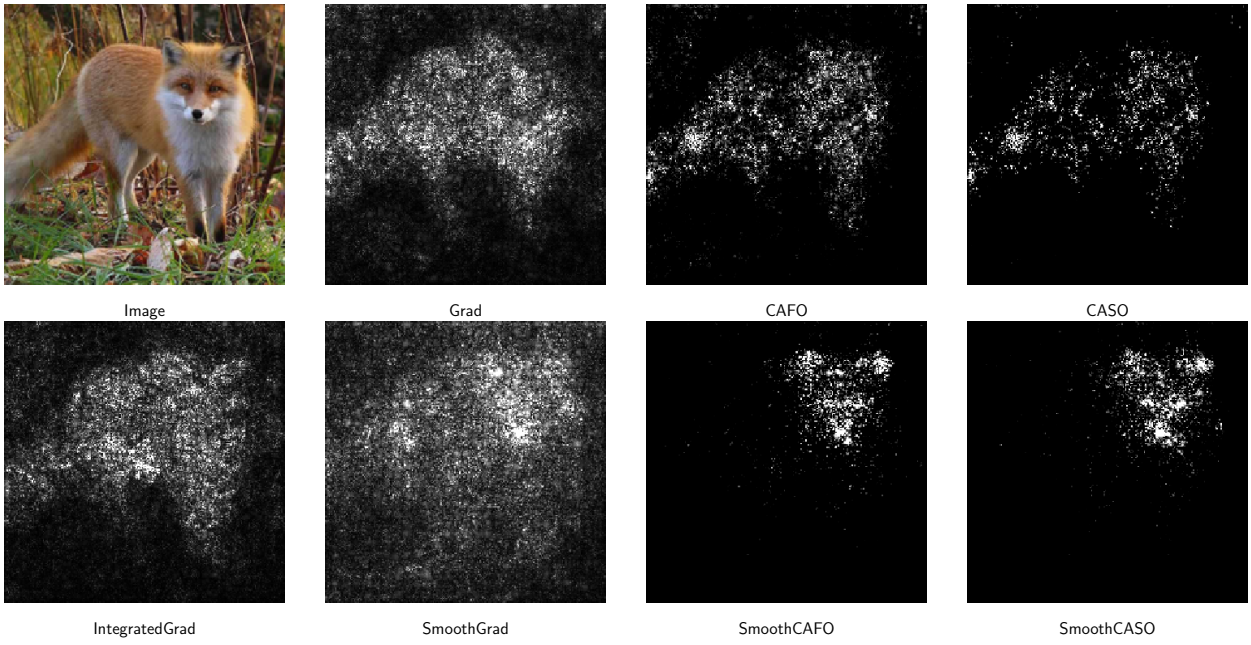


Figure 8.

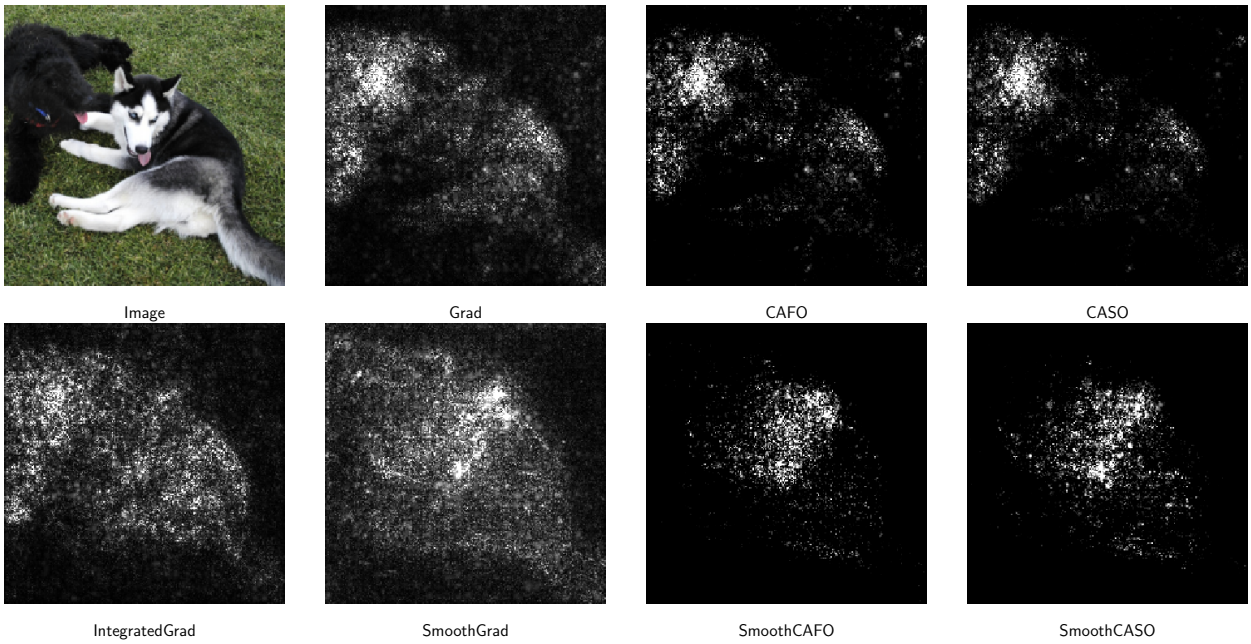


Figure 9.

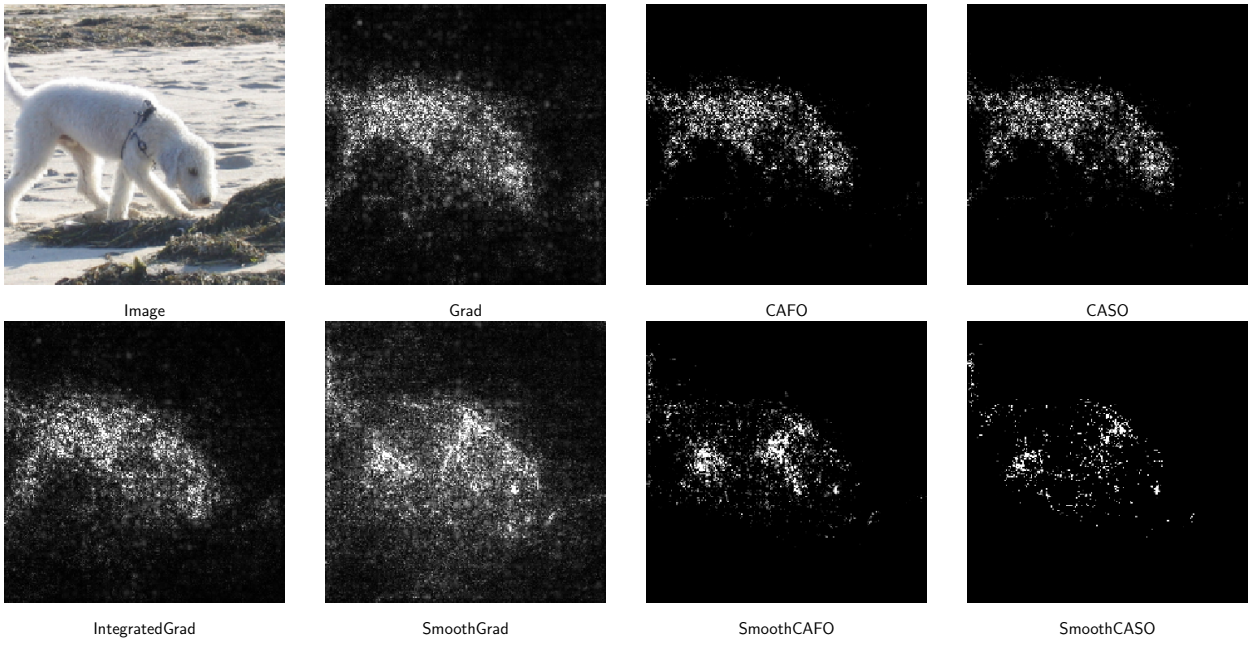


Figure 10.