Appendix
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]

A. Proofs

A.1. Proof of Proposition 1

This section derives the closed-form formula for the Hessian of the loss function for 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

$$\frac{\partial \hat{\mathbf{y}}_i}{\partial \mathbf{x}} = \mathbf{W}_i \tag{1}$$

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

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

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

$$\implies \frac{\partial \ell(\mathbf{p}, \mathbf{y})}{\partial \hat{\mathbf{y}}_i} = \mathbf{p}_i - \mathbf{y}_i \tag{2}$$

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

Using (1) and (2)

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

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

Therefore, we have:

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

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

$$\mathbf{H}_{\mathbf{x}} = \sum_{i=1}^{c} \mathbf{W}_i(\nabla_{\mathbf{x}} \mathbf{p}_i)^T \tag{3}$$

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

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

$$\implies \nabla_{\mathbf{x}} \mathbf{p}_i = \sum_{j=1}^{c} \left(\mathbf{W}_j \times \frac{\partial \mathbf{p}_i}{\partial \hat{\mathbf{y}}_j} \right) \quad \text{(Using (1))} \tag{4}$$

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

$$\implies \nabla_{\hat{\mathbf{y}}} \mathbf{p} = \text{diag}(\mathbf{p}) - \mathbf{p}\mathbf{p}^T \tag{5}$$

$$\mathbf{H}_{\mathbf{x}} = \sum_{i=1}^{c} \mathbf{W}_i \left(\sum_{j=1}^{c} \mathbf{W}_j \times \frac{\partial \mathbf{p}_i}{\partial \hat{\mathbf{y}}_j} \right)^T \quad \text{(Substituting (4) in (3))}$$

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

$$\implies \mathbf{H}_{\mathbf{x}} = \mathbf{W}(\text{diag}(\mathbf{p}) - \mathbf{p}\mathbf{p}^T)\mathbf{W}^T \quad \text{(Using (5))}$$

Thus we have,

$$\nabla_{\mathbf{x}} \ell(\mathbf{p}, \mathbf{y}) = \mathbf{g}_{\mathbf{x}} = \mathbf{W}(\mathbf{p} - \mathbf{y}) \tag{6}$$

$$\mathbf{H}_{\mathbf{x}} = \mathbf{W}\mathbf{A}\mathbf{W}^T \tag{7}$$

where

$$\mathbf{A} := \text{diag}(\mathbf{p}) - \mathbf{p}\mathbf{p}^T. \tag{8}$$

This completes the proof.

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

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

A.2. Proof of Theorem 2

To simplify notation, define $\mathbf{A}$ as in (8). For any arbitrary row of the matrix $\mathbf{A}_i$, we have

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

Because $|\mathbf{A}_{ii}| >= \sum_{j\neq i}|\mathbf{A}_{ij}|$, 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 positive semidefinite, we can write $\mathbf{A} = \mathbf{L}\mathbf{L}^T$. Using (7):

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

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:

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

where $\Delta := \tilde{\mathbf{x}} - \mathbf{x}$ (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 (6) and (7) respectively. Without loss of generality, assume that the first class is the class with maximum probability. Hence,

$$\mathbf{y} = [1, 0, 0, ..., 0]^T. \tag{9}$$

We assume all other classes have small probability (i.e., the confidence is high),

$$\mathbf{p}_i = \epsilon \approx 0 \quad \forall\ \mathrm{i} \in [2, \mathrm{c}]$$

Since $\sum_{i=1}^{c}\mathbf{p}_i = 1$,

$$\begin{aligned}
&\implies \mathbf{p}_1 = 1 - (c-1)\epsilon,\\
&\implies \mathbf{p} = [1-(c-1)\epsilon,\ \epsilon, \ldots,\ \epsilon]^T
\end{aligned} \tag{10}$$

We define:

$$\mathbf{A} = \mathrm{diag}(\mathbf{p}) - \mathbf{p}\mathbf{p}^T$$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1c}\\ a_{21} & a_{22} & \ldots & a_{2c}\\ \vdots & \vdots & \ddots & \vdots\\ a_{c1} & a_{c2} & \ldots & a_{cc}\end{bmatrix}$$

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

Ignoring ϵ^2 terms:

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

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:

$$(c-1)\epsilon v_1 - \epsilon\sum_{i=2}^{c}v_i = \lambda v_1 \tag{11}$$

$$-\epsilon v_1 + \epsilon v_i = \lambda v_i \ \forall i \in [2, c]$$

$$\implies v_i = \frac{\epsilon}{\epsilon-\lambda}v_1 \ \forall i \in [2, c], \text{ for } \lambda \neq \epsilon \tag{12}$$

$$\implies \text{or } v_1 = 0, \text{ for } \lambda = \epsilon \tag{13}$$

We first consider the case $\lambda \neq \epsilon$ (12). Substituting v_i in (11):

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

$$\begin{aligned}
&(c-1)\epsilon v_1\left[1 - \frac{\epsilon}{\epsilon-\lambda}\right] = \lambda v_1\\
&(c-1)\epsilon v_1\left[-\frac{\lambda}{\epsilon-\lambda}\right] = \lambda v_1\\
&(c-1)\epsilon v_1(-\lambda) = \lambda v_1(\epsilon-\lambda)\\
&\implies \lambda v_1(c\epsilon - \lambda) = 0\\
&\implies \lambda = 0 \text{ or } v_1 = 0 \text{ or } \lambda = c\epsilon\\
&v_1 = 0 \implies v_i = \frac{\epsilon}{\epsilon-\lambda}v_1 = 0 \quad \forall\ i \in [2, c]\\
&\implies \mathbf{v} = 0
\end{aligned}$$

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 (12):

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

Dividing by the normalization constant,

$$\mathbf{u}_1^T = \frac{1}{\sqrt{c(c-1)}}[1-c, 1, ..., 1] \tag{14}$$

Now we consider the case $\lambda = \epsilon$ (13). Substituting $v_1 = 0$, $\lambda = \epsilon$ in (11):
The space of eigenvectors for $\lambda = \epsilon$ is a $c-2$ 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

$$\mathbf{A}_1 = c\epsilon\mathbf{u}_1\mathbf{u}_1^T, \quad \mathbf{A}_2 = \epsilon\sum_{i=2}^{c-1}\mathbf{u}_i\mathbf{u}_i^T$$

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

Hence as $c \to \infty$,

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

Using (7),

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

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

$$\mathbf{H}_\mathbf{x} \approx c\epsilon\mathbf{W}\mathbf{u}_1\mathbf{u}_1^T\mathbf{W}^T \tag{15}$$

Using (6),

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

Let $\mathbf{W}_i$ denote the i^{th} row of $\mathbf{W}$,
Using (9) and (10),

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

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

Using (14),

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

$$\implies \mathbf{W}\mathbf{u}_1 = \frac{\mathbf{g}_\mathbf{x}}{\epsilon\sqrt{c(c-1)}} \tag{16}$$

Using (15),

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

Using (16),

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

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

$$\implies \mathbf{H}_\mathbf{x} \approx \frac{\mathbf{g}_\mathbf{x}\mathbf{g}_\mathbf{x}^T}{\epsilon(c-1)} \tag{17}$$

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 $\mathbf{g}_\mathbf{x} = \nabla_\mathbf{x}\ell(\mathbf{p}, \mathbf{y}) = \mathbf{W}(\mathbf{p} - \mathbf{y})$ (6).
Let $\lambda_1 = 0$ in the CASO and CAFO objectives. The CASO objective then 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 (17),

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

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:

$$\mathbf{z}_1 = \frac{\mathbf{g}_\mathbf{x}}{\|\mathbf{g}_\mathbf{x}\|}$$

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 (Definition 3). Let $\Delta^{(t)}$ be the value of Δ in the t^{th} step with a learning rate $\alpha \leq \lambda_2 - L/2$. We have*

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

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

By Theorem 2, the Cholesky decomposition of $\mathbf{A}$ (defined in (8)) 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 from SmoothGrad (Smilkov et al., 2017). This normalization method first takes the absolute value of the importance scores and then 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 $[0, 1]$. Code for the method is available.[1]

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, $y = W\mathbf{x}$, multiplying 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) argue similarly 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 (Kindermans et al., 2018). We observed numerous failures in existing interpretation methods when input multiplication is removed.

[1] 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:

$$\min_{\Delta} \quad \|\mathbf{y} - \mathbf{A}\Delta\|^2, \qquad \|\Delta\|_0 \leq k, \tag{18}$$

where

$$\mathbf{A} := \left(\lambda_2 \mathbf{I} - \frac{1}{2}\mathbf{H}_{\mathbf{x}}\right)^{1/2} \tag{19}$$

$$\mathbf{y} := \frac{1}{2}\mathbf{A}^{-1}\nabla_{\mathbf{x}}\ell\left(f_{\theta^*}(\mathbf{x}), y\right). \tag{20}$$

Where $(.)^{1/2}$ indicates the square root of a positive definite matrix. Equation (19) 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:

$$\mathcal{C}(S; \alpha) := \{\Delta \in \mathbb{R}^d : \|\Delta_{S^c}\|_1 \leq \|\Delta_S\|_1\}, \tag{21}$$

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

$$\frac{1}{d}\|\mathbf{A}\Delta\|_2^2 \geq \gamma^2 \|\Delta\|_2^2 \quad \forall \Delta \in \mathcal{C}(S; \alpha). \tag{22}$$

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 (18) is tight (Bickel et al., 2009). In other words, if Δ^* is the solution of optimization (18), it is also the solution of optimization

$$\min_{\Delta} \quad \|\mathbf{y} - \mathbf{A}\Delta\|^2, \qquad \|\Delta\|_1 \leq \|\Delta^*\|_1. \tag{23}$$

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, our empirical experiments are consistent with our theory: the resulting Δ of the CASO interpretation objective is sparse for proper choices of the regularization parameters.

F. Empirical Analysis of the Hessian Impact

F.1. Empirically Verifying the Hessian Approximation

Theorems 4 and 5 are valid only in the asymptotic regime. A similar analysis in the finite regime is more challenging as it requires the use of perturbation analysis of matrix eigenvalues and eigenvectors. However, we can do a simulation to assess the rate of convergence of the Hessian matrix to such a rank one matrix as the number of classes increases (in Figure 1) and the probability of the predicted class tends to 1 (in Figure 2).

For the simulation, we create a linear model $\hat{\mathbf{y}} = \mathbf{W}^T\mathbf{x} + \mathbf{b}$ where $\mathbf{W}$ and $\mathbf{b}$ are initialized to random values. Since Theorem 4 does not assume a trained network, our analysis is valid even with randomly initialized values of $\mathbf{W}$ and $\mathbf{b}$. 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 (6) and (7), respectively. Without loss of generality, assume that the first class is the one with maximum probability. Thus we create a probability vector $\mathbf{p} = [1 - (c-1)\epsilon,\ \epsilon,\ \ldots,\ \epsilon]$ and a prediction vector $\mathbf{y} = [1.,\ 0.,\ \ldots,\ 0.]$. Using Proof A.4, we have that the Hessian of this model can be approximated using:

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

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

For Figure 1, we fix the probability of predicted class to be 0.9999 (we call it $\mathbf{p}[0]$) and vary c from 10 to 1000. Thus, note that ϵ varies as c varies and is given by $\epsilon = \frac{1-\mathbf{p}[0]}{c-1}$. For Figure 2, we fix number of classes to be 100 and vary ϵ from $5e-3$ to $1e-6$ (an interval length of $1e-6$). Similarly, Figure 3 shows a comparison between the relative error as the number of classes and probability are both varied. We observe that the relative error converges quickly as a function of both the number of classes and -log(1-$\mathbf{p}[0]$).

F.2. Comparing CASO and CAFO for ReLU networks

We show additional examples (for relu networks) with low confidence in the predicted class in Figure 4. The interpretations produced by CASO and CAFO are qualitatively different.

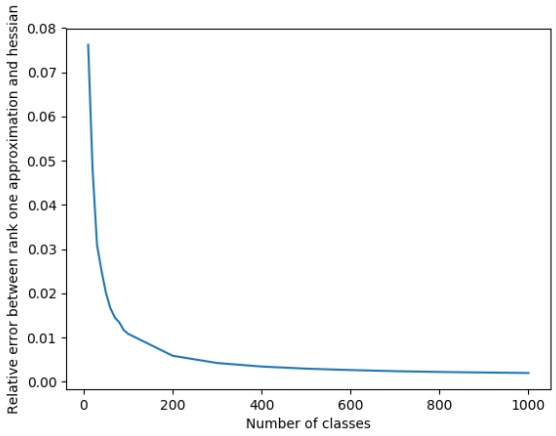

Figure 1. The relative error between a rank one approximation of the Hessian and the true Hessian as the number of classes increases. Although our theoretical analysis only holds in the asymptotic regime, the Hessian's convergence to a rank one matrix happens quicky empirically.

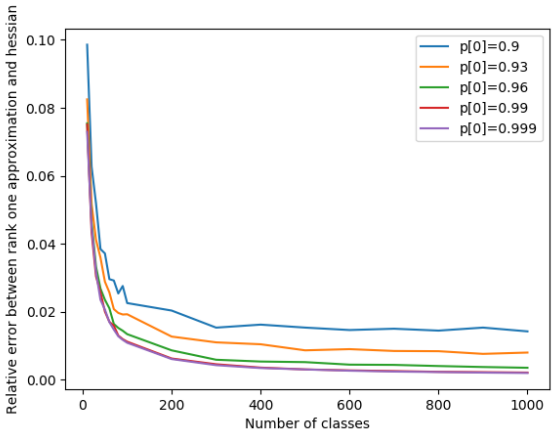

Figure 3. The relative error between a rank one approximation of the Hessian and the true Hessian when varying the number of classes and the probability of predicted class (denoted by p[0]).

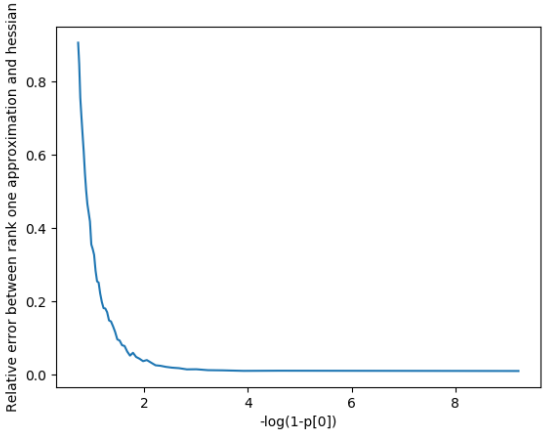

Figure 2. The relative error between a rank one approximation of the Hessian and the true Hessian as the probability of the predicted class grows. We use a log scale and denote the predicted probability as p[0].

F.3. Experiments with General Non-linearities

We use a SE-Resnet-50 (Hu et al., 2018), a neural network with sigmoid non-linearities. The sigmoid non-linearity causes the model to no longer be piecewise linear. We generate saliency maps using the same 1000 random samples as in Section 3.3.

We plot the Frobenius norm of the difference between CASO and CAFO in Figure 6. We normalize the solutions produced by CASO and CAFO to have the same L_2 norm before taking the difference. Even though the model is no longer piecewise linear, the empirical results are consistent with Theorem 5 (which only holds for piecewise linear networks).

To observe the difference between CAFO and CASO interpretations, we compare them for two images with low classification confidence in Figure 5. The interpretations produced by CASO and CAFO are qualitatively different.

G. Additional Details on Experiments

G.1. Details on Experiments Reported in Figure 1

Current autograd software does not support fast eigenvalue decomposition of a matrix in a batched setting. This makes computing the exact hessian inefficient when interpreting numerous samples. 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.

G.2. Details on Experiments Reported in Figure 3

Details of the hyperparameters used in Figure 3 are shown in Table 2.

G.3. Details on Experiments Reported in Figure 4

Details of the hyperparameters used in Figure 4 are shown in Table 3.

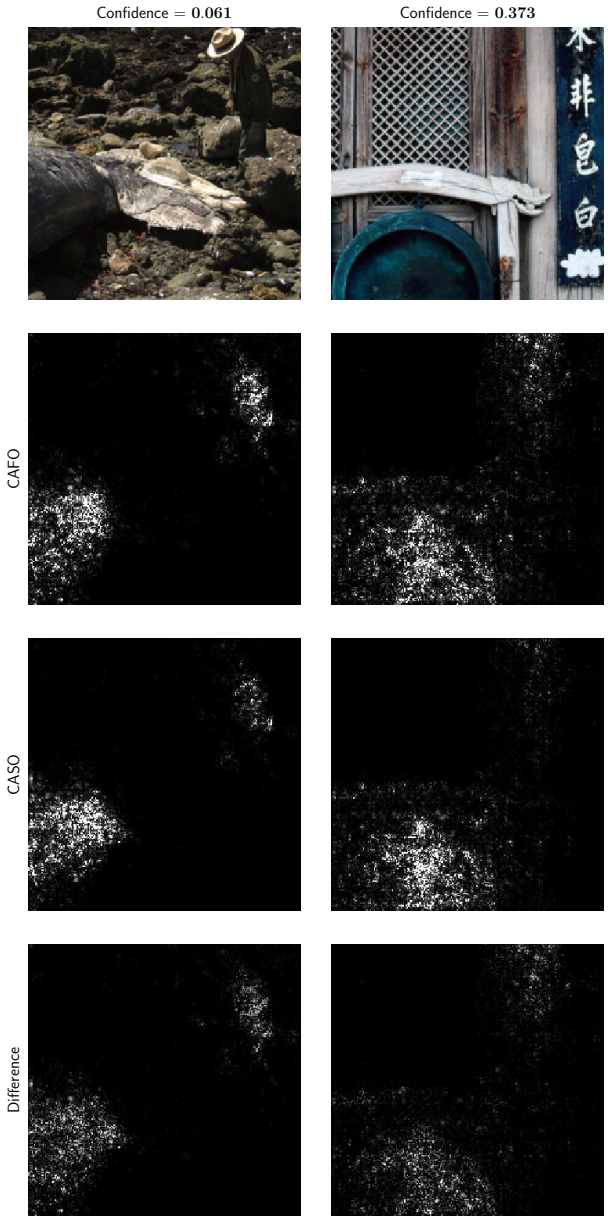

Figure 4. CASO and CAFO interpretations for low confidence examples for a network with ReLU activations.

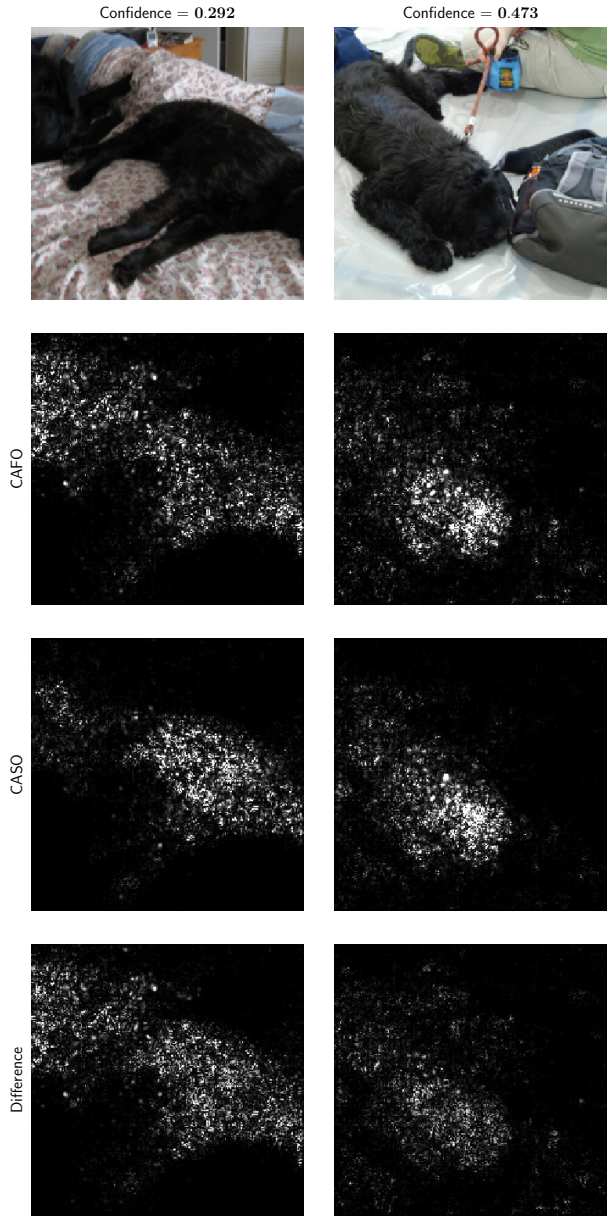

Figure 5. CASO and CAFO interpretations for low confidence predictions using a model that is not piecewise linear (SE-Resnet-50).

H. Comparison with existing methods

Figures 7–12 provide further examples of our interpretation method with existing techniques.

References

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.

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.

Bickel, P. J., Ritov, Y., Tsybakov, A. B., et al. Simultaneous

Figure 6. Scatter plot showing the Frobenius norm difference between CASO and CAFO (after normalizing both vectors to have the same L_2 norm) for a network which is not piecewise linear (SE-Resnet-50).

Table 1. Hyper-parameter details for Figure 1

Parameter	Configuration
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

Table 2. Hyper-parameter details for Figure 3

Parameter	Configuration
λ_1 values	$0, 10^{-5}, 10^{-4}, 10^{-3}, 6.25\times10^{-3}, 1.25\times10^{-2}, 2.5\times10^{-2}, 5\times10^{-2}$
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

Table 3. Hyper-parameter details for Figure 4

Parameter	Configuration
λ_1 values	$0, 10^{-5}, 10^{-4}, 10^{-3}, 6.25\times10^{-3}, 1.25\times10^{-2}, 2.5\times10^{-2}, 5\times10^{-2}$
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

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.

Hu, J., Shen, L., and Sun, G. Squeeze-and-excitation networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

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. *Neural Information Processing Systems*, 2018.

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.

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.

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

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

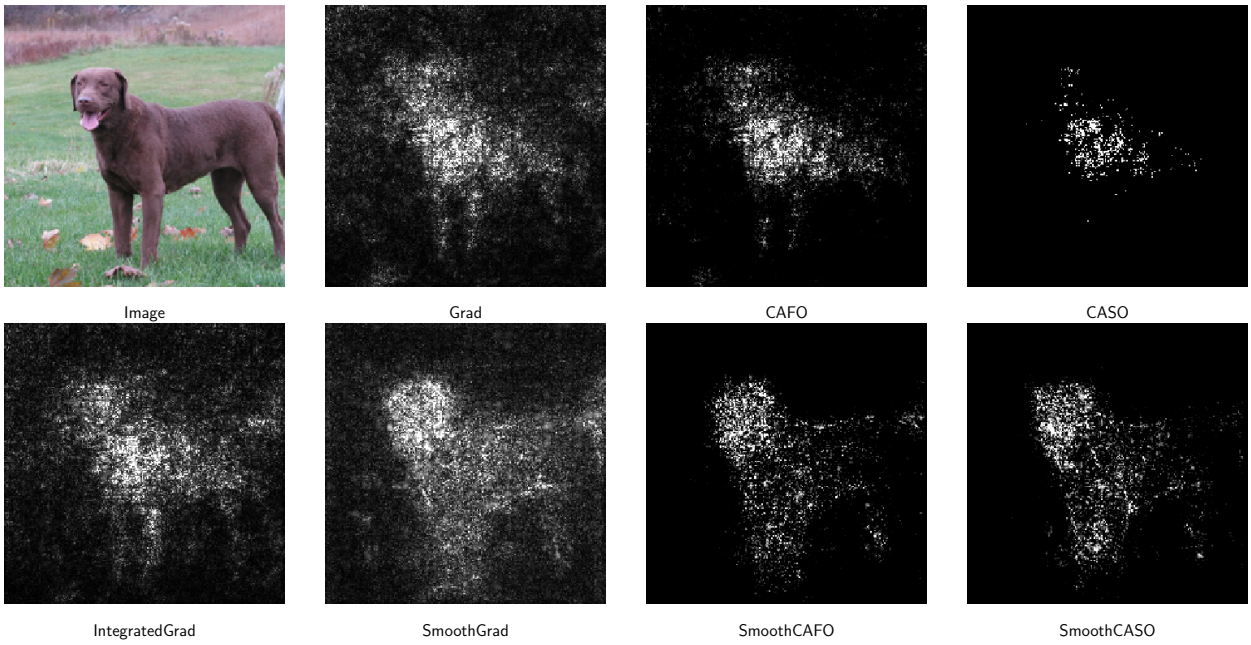

Figure 7.

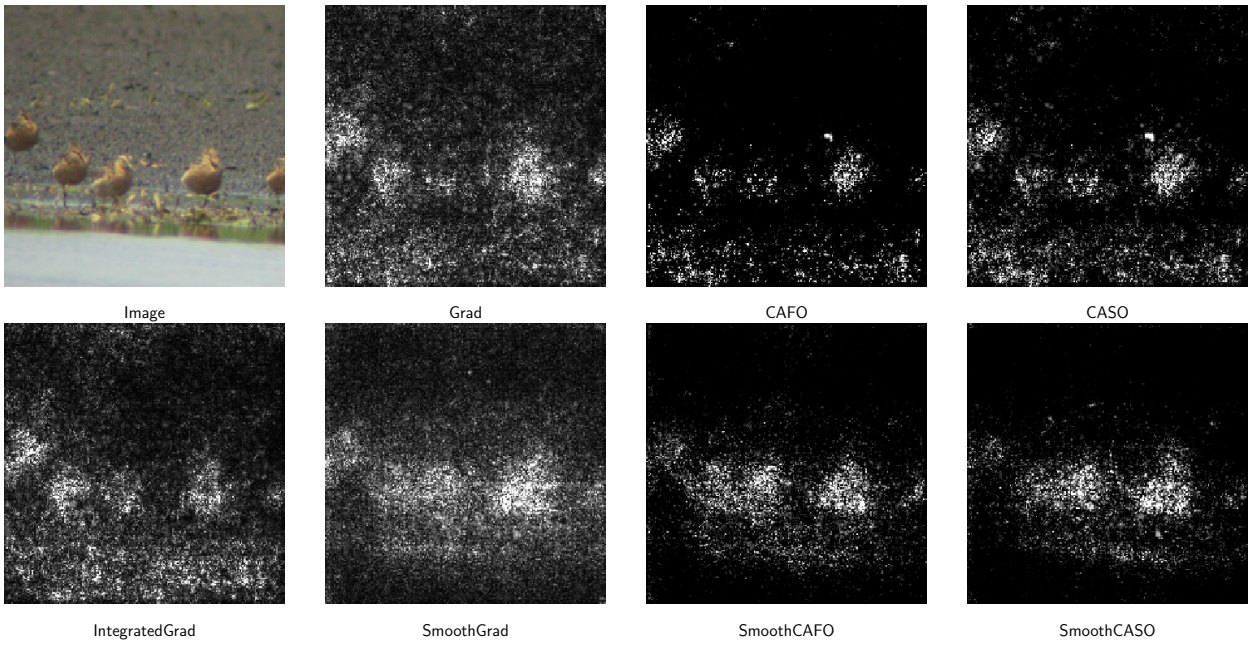

Figure 8.

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

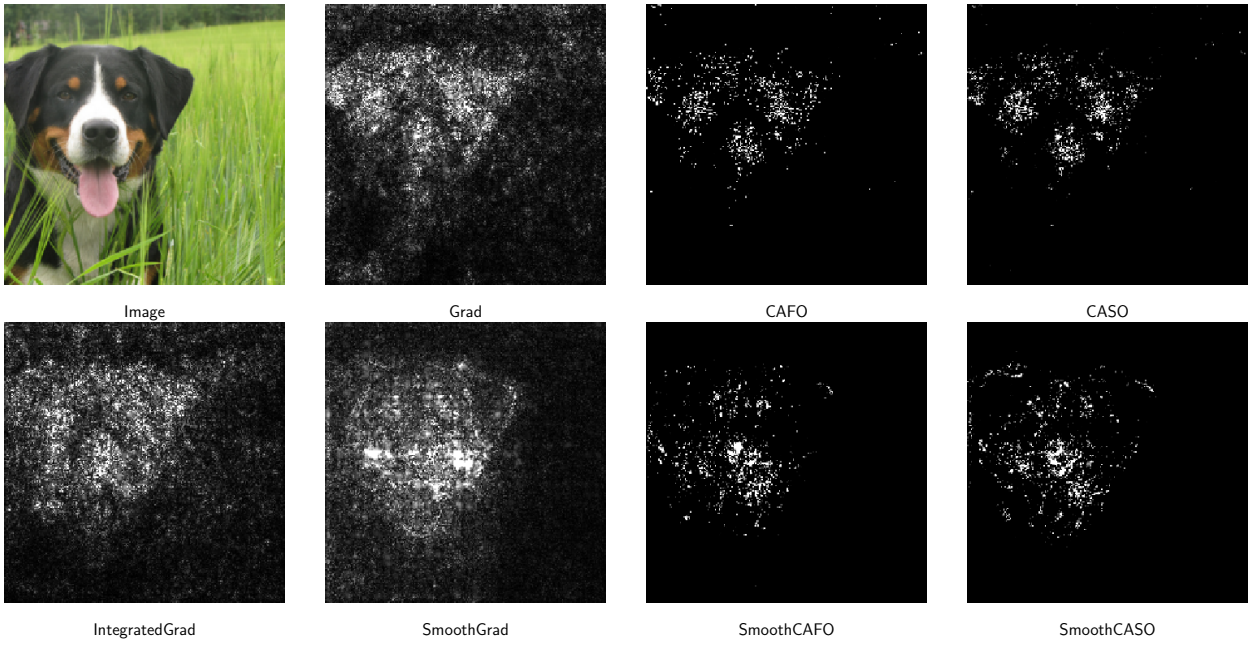

Figure 9.

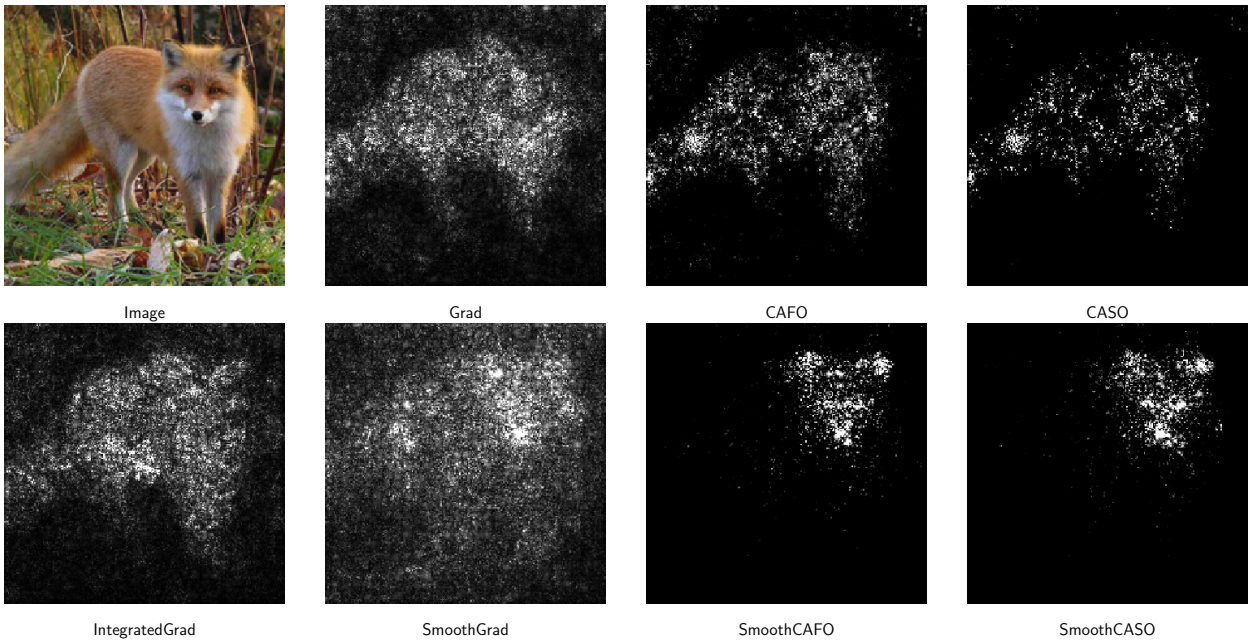

Figure 10.

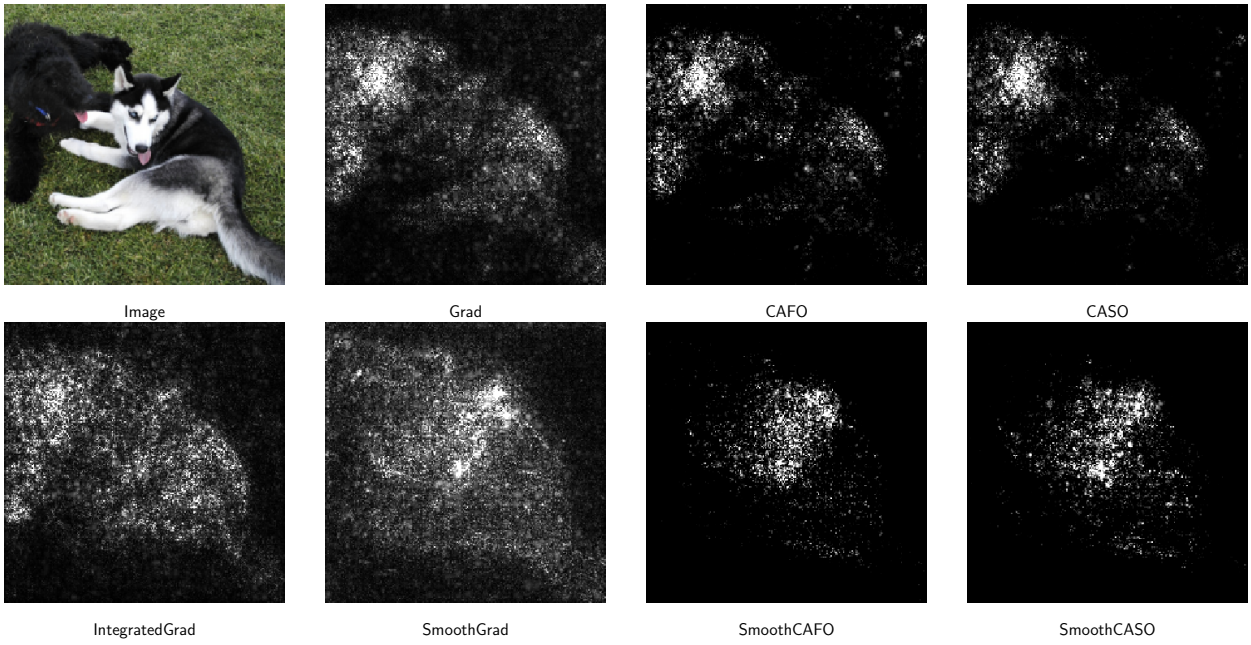

Figure 11.

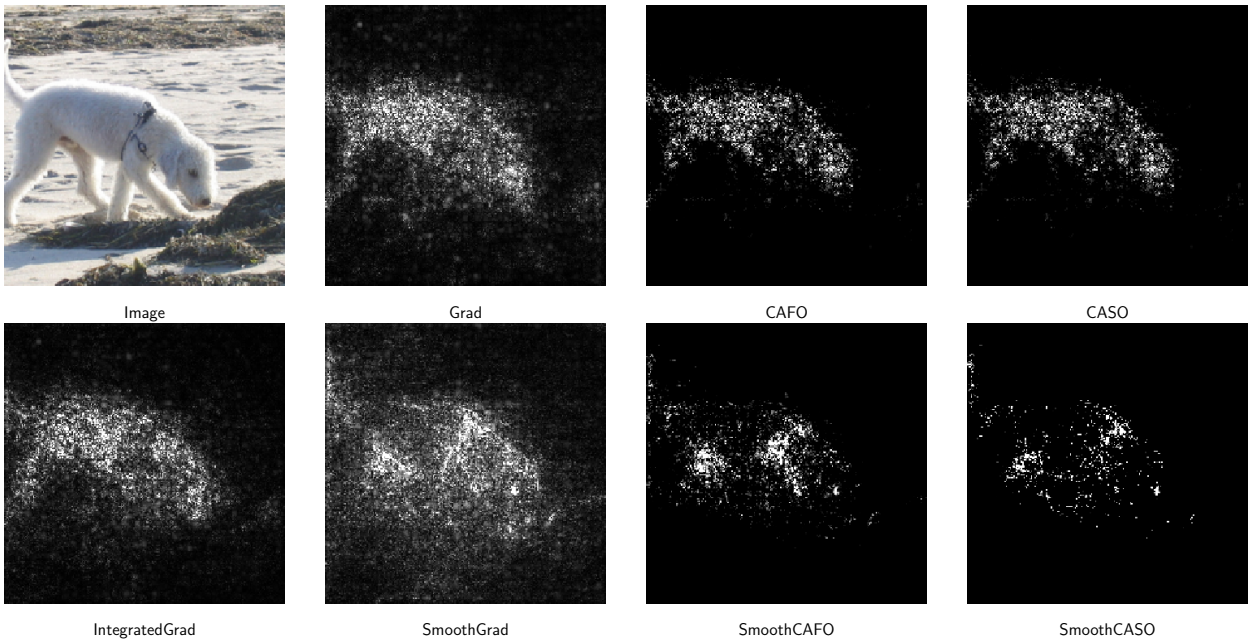

Figure 12.