Lecture 8. Explaining Neural Networks

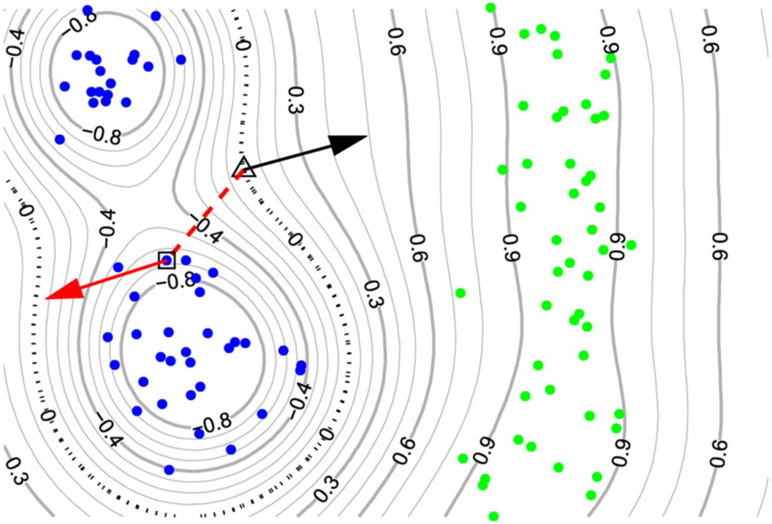
DeepLIFT · Integrated Gradients · SmoothGrad · GradCAM

Tatiana Makhalova
MADE, 06.11.2021

Recap: first order Taylor approximation

The first order Taylor approximation $f(x) \approx f(x_0) + \sum_{d=1}^{V} f'(x_0)(x_{(d)} - x_{0(d)})$

We are interested to find out the contribution of each pixel relative to the state of maximal uncertainty of the prediction, i.e., $f(x_0) = 0$, i.e., $f(x) \approx \sum_{d=1}^{V} f'(x_0)(x_{(d)} - x_{0(d)})$



Δ the nearest root point x_0 on the decision boundary

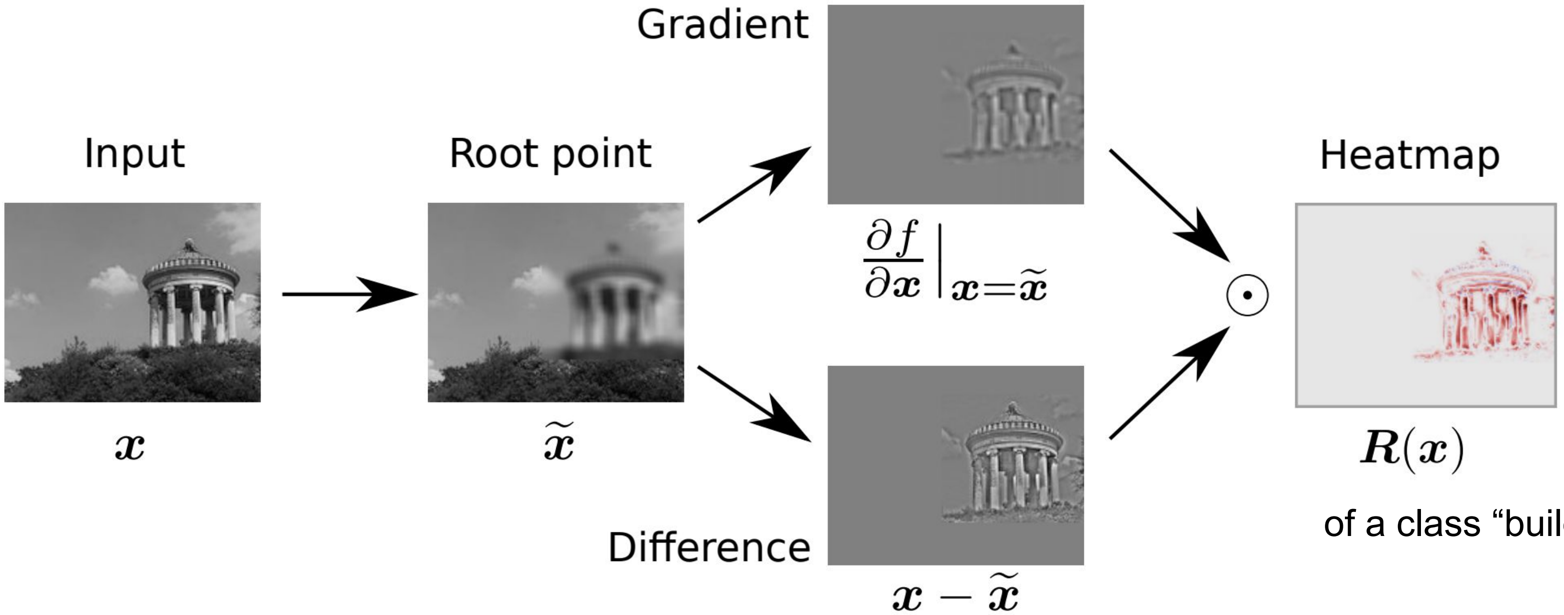
$f'(x_0)$

$x - x_0$

the approximation of $f(x)$ by Taylor expansion around x_0 (equivalent to the diagonal of the outer product between $f'(x_0)$ and $x - x_0$)

Bach, Sebastian, et al. "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation." PloS one 10.7 (2015): e0130140.

Intuitive example for the Deep Taylor Decomposition



of a class “building”

it is needed to find a “good’ root point such that the concept (to be classified on the image) is the most prominent when the Taylor decomposition is applied. The gradient measures the **sensitivity** of the class “building” to each pixel when the classifier f is evaluated at the root point, the difference is the image containing only an object “building”

Montavon, Grégoire, et al. "Explaining nonlinear classification decisions with deep taylor decomposition." Pattern Recognition 65 (2017): 211-222

Examples of reference (root) points

For a given instance X containing an object of a certain class C, there exists many ways to “remove” this object from the given instance X, e.g.,

- **background of images** for simple tasks (as the digit recognition in MNIST)
- **blurred version of the images** for object classification (computer vision)
- the **expected frequencies of ACGT** in the background for DNA classification

Deep Learning Important FeaTures. Motivation

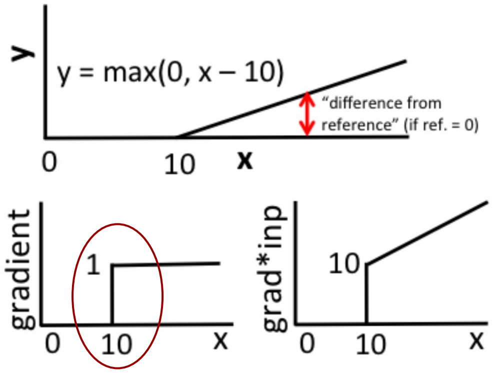
Gradients, DecovNet, Guided backpropagation do not use a reference point (image)

LRP, DeepTaylor, PatternAttribution use (implicitly) **various reference (root) points** (specific for each neuron)

DeepLIFT exploits a specific reference point and improves some issues of the previous methods

Shrikumar, Avanti, Peyton Greenside, and Anshul Kundaje. "Learning important features through propagating activation differences." International Conference on Machine Learning. PMLR, 2017

When do the gradient-based approaches fail?



$$f(x) = max(0, x - 10)$$

Discontinuous gradients can produce misleading importance scores

Both "**gradient**" and "**gradient × input**" have a discontinuity at $x = 10$

At $x = 10 + \varepsilon$: for "**gradient × input**" the attribution is given by 1 * (10 + ε − 10), i.e., the contribution of x is 10 + ε, the contribution of the bias is -10

At $x < 10$, contributions of x and the bias term are both 0

The difference-from-reference $f(x) - f(0)$ (red arrow, top figure) gives a continuous increase in the contribution score

DeepLIFT

Let $x_1, x_2, \ldots, x_n$ represents the activations of some neurons in an intermediate layer for an input I, and $x_1^0, x_2^0, \ldots, x_n^0$ represent the activations of the same neurons at the **reference point** I^0

Let t be the activation of the output layer at I, and t^0 be the activation of the output layer at I

$\Delta t = t - t^0$ is the **difference-from-reference**

The contribution scores $C_{\Delta x i \Delta t}$ for Δx_i are defined s.t.:

$$\sum_{i=1}^{n} C_{\Delta x_i \Delta t} = \Delta t$$

DeepLIFT multipliers and their properties

Multipliers are defined as follows:

$$m_{\Delta x \Delta t} = \frac{C_{\Delta x \Delta t}}{\Delta x}$$

They are close to partial derivatives except for instead of infinitesimal changes for t and x we consider the finite ones

For the multipliers the **chain rule** holds, i.e.

$$m_{\Delta x_i \Delta t} = \sum_j m_{\Delta x_i \Delta y_j} m_{\Delta y_j \Delta t}$$

DeepLIFT propagation rules

The negative and positive contributions are considered separately:

$$\Delta y = \Delta y^{+} + \Delta y^{-}$$
$$C_{\Delta y \Delta t} = C_{\Delta y^{+} \Delta t} + C_{\Delta y^{-} \Delta t}$$

DeepLIFT proposes different rules for different components of neural networks:

- **linear** rule for linear function
- **rescale** or **reveal cancel** rule for non-linear rules
- a special correction for softmax

Linear rule

For a linear function $y = \sum_i w_i x_i + b$, the difference is $\Delta y = \sum_i w_i \Delta x_i$

$$\begin{aligned}\Delta y^+ &= \sum_i 1\{w_i \Delta x_i > 0\} w_i \Delta x_i \\ &= \sum_i 1\{w_i \Delta x_i > 0\} w_i (\Delta x_i^+ + \Delta x_i^-)\end{aligned}$$

$$\begin{aligned}\Delta y^- &= \sum_i 1\{w_i \Delta x_i < 0\} w_i \Delta x_i \\ &= \sum_i 1\{w_i \Delta x_i < 0\} w_i (\Delta x_i^+ + \Delta x_i^-)\end{aligned}$$

The contributions are given by

$$\begin{aligned}C_{\Delta x_i^+ \Delta y^+} &= 1\{w_i \Delta x_i > 0\} w_i \Delta x_i^+ \\ C_{\Delta x_i^- \Delta y^+} &= 1\{w_i \Delta x_i > 0\} w_i \Delta x_i^- \\ C_{\Delta x_i^+ \Delta y^-} &= 1\{w_i \Delta x_i < 0\} w_i \Delta x_i^+ \\ C_{\Delta x_i^- \Delta y^-} &= 1\{w_i \Delta x_i < 0\} w_i \Delta x_i^-\end{aligned}$$

Linear rule

Combining $m_{\Delta x \Delta t} = \frac{C_{\Delta x \Delta t}}{\Delta x}$ and
$$\begin{aligned}
C_{\Delta x_i^+ \Delta y^+} &= 1\{w_i \Delta x_i > 0\} w_i \Delta x_i^+ \\
C_{\Delta x_i^- \Delta y^+} &= 1\{w_i \Delta x_i > 0\} w_i \Delta x_i^- \\
C_{\Delta x_i^+ \Delta y^-} &= 1\{w_i \Delta x_i < 0\} w_i \Delta x_i^+ \\
C_{\Delta x_i^- \Delta y^-} &= 1\{w_i \Delta x_i < 0\} w_i \Delta x_i^-
\end{aligned}$$

we get the multipliers
$$\begin{aligned}
m_{\Delta x^+ \Delta y^+} &= m_{\Delta x^- \Delta y^+} = 1\{w_i \Delta x_i > 0\} w_i \\
m_{\Delta x^+ \Delta y^-} &= m_{\Delta x^- \Delta y^-} = 1\{w_i \Delta x_i < 0\} w_i
\end{aligned}$$

If $\Delta x_i = 0$, it does not imply $\Delta x_i^+ = 0$, nor $\Delta x_i^- = 0$. In order to be able to propagate to such neurons the relevance we use the following rule $m_{\Delta x^+ \Delta y^+} = m_{\Delta x^+ \Delta y^-} = 0.5 w_i$

Rescale rule

For non-linear transformations with a single output $y = f(x)$, e.g., ReLU, tanh, sigmoid)

The **contribution**: $C_{\Delta x \Delta y} = \Delta y$, the **multipliers** $m_{\Delta x \Delta y} = \frac{\Delta y}{\Delta x}$

For positive and negative components:

$$\Delta y^{+} = \frac{\Delta y}{\Delta x} \Delta x^{+} = C_{\Delta x^{+} \Delta y^{+}}$$

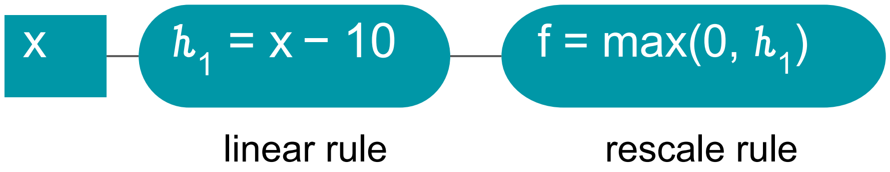
$$\Delta y^{-} = \frac{\Delta y}{\Delta x} \Delta x^{-} = C_{\Delta x^{-} \Delta y^{-}}$$

$$m_{\Delta x^{+} \Delta y^{+}} = m_{\Delta x^{-} \Delta y^{-}} = m_{\Delta x \Delta y} = \frac{\Delta y}{\Delta x}$$

When $x - x_0 < \varepsilon$ we use $\frac{dy}{dx}$ to avoid numerical instability caused by a very small denominator

Example 1

$$f(x) = max(0, x - 10)$$



0. Initialisation

Let a reference point be x = 0 then h_1 = -10 and f = 0

Consider a point be x = ε + 10 then h_1 = ε and f = ε

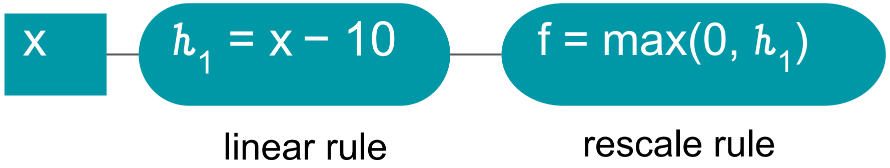
1. Linear rule

$m_{\Delta x^+ \Delta h_1^+} = 1$ (note that $m_{\Delta x^- \Delta h_1^+}$, $m_{\Delta x^- \Delta h_1^-}$, and $m_{\Delta x^+ \Delta h_1^-}$ do not exist, only one component is possible)

$$m_{\Delta x^+ \Delta h_1^+} \cdot \Delta x = C_{\Delta x^+ \Delta h_1^+} = 1 \cdot (\varepsilon + 10) = \varepsilon + 10$$

Example 1

$$f(x) = max(0, x - 10)$$



2. Rescale rule

$$m_{\Delta h_1^+ \Delta f^+} = m_{\Delta h_1^- \Delta f^-} = m_{\Delta h_1 \Delta f} = \frac{\Delta f}{\Delta h_1}$$

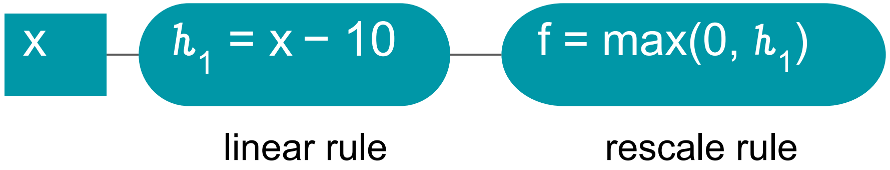
$$m_{\Delta h_1^+ \Delta f^+} = \frac{\varepsilon}{\varepsilon - (-10)} = \frac{\varepsilon}{\varepsilon + 10}$$

$$C_{\Delta h_1^+ \Delta f^+} = m_{\Delta h_1^+ \Delta f^+} \cdot \Delta h_1^+ = \frac{\varepsilon}{\varepsilon - (-10)} \cdot (\boxed{\varepsilon} - \boxed{(-10)}) = \varepsilon$$

$$h_1(\varepsilon + 10) = \varepsilon \quad h_1(0) = -10$$

Example 1

$$f(x) = max(0, x - 10)$$



3. Chain rule

$$m_{\Delta x \Delta f} = m_{\Delta x \Delta h_1} \cdot m_{\Delta h_1 \Delta f}$$

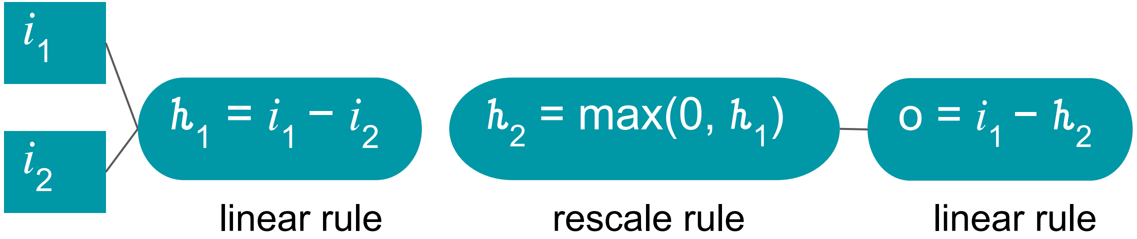
$$m_{\Delta x \Delta f} = 1 \cdot \frac{\varepsilon}{\varepsilon + 10}$$

4. Contribution

$$C_{\Delta x \Delta f} = m_{\Delta x \Delta f} \cdot \Delta x = \frac{\varepsilon}{\varepsilon + 10} \cdot (\varepsilon + 10) = \varepsilon$$

Example 2

$o = min(i_1, i_2)$



0. Initialisation

Let a reference point be $i_1 = i_2 = 0$ then $h_1 = 0$, $h_2 = 0$, $o = 0$

Consider points $i_1 > i_2 > 0$ then $h_1 = i_1 - i_2$, $h_2 = i_1 - i_2$, $o = i_1 - (i_1 - i_2) = i_2$

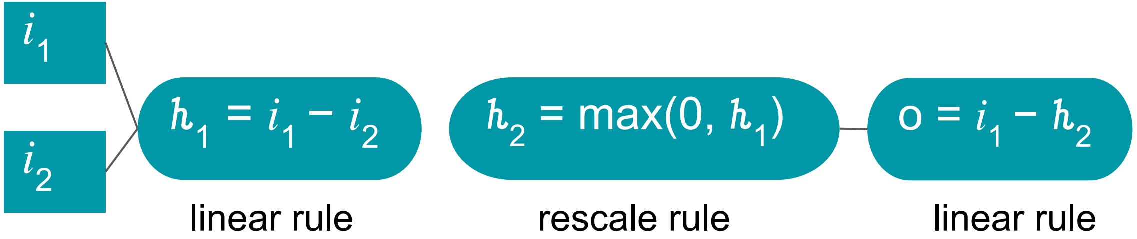
1. Linear rule

$$m_{\Delta i_1^+ \Delta h_1^+} = 1 \qquad C_{\Delta i_1^+ \Delta h_1^+} = i_1 \Rightarrow \Delta h_1^+ = i_1$$

$$m_{\Delta i_2^+ \Delta h_1^-} = -1 \quad C_{\Delta i_2^+ \Delta h_1^-} = -i_2 \Rightarrow \Delta h_1^- = -i_2$$

Example 2

$o = min(i_1, i_2)$



2. Rescale rule (no difference between Δh_1^+ and Δh_1^-)

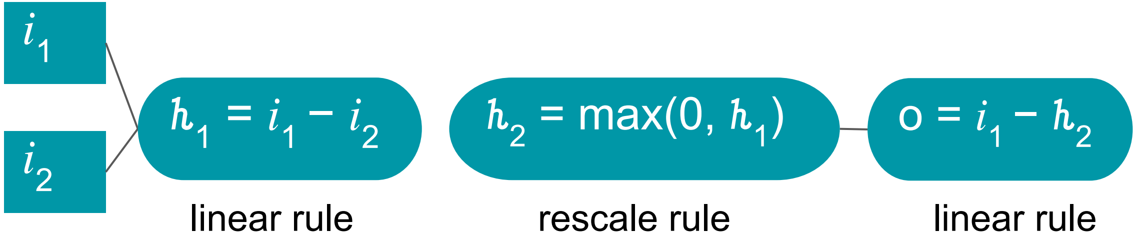
$$\Delta h_1 = \Delta h_1^+ + \Delta h_1^- = i_1 - i_2$$

$$m_{\Delta h_1 \Delta h_2} = \frac{\Delta h_2}{\Delta h_1} = \frac{(i_1 - i_2) - 0}{(i_1 - i_2) - 0} = 1$$

$$C_{\Delta h_1 \Delta h_2} = m_{\Delta h_1 \Delta h_2} \cdot \Delta h_1 = 1 \cdot (i_1 - i_2) = i_1 - i_2$$

Example 2

$$o = min(i_1, i_2)$$



2. Linear rule

$$m_{\Delta i_1 \Delta o} = 1 - m_{\Delta i_1 \Delta h_2} \boxed{=} 1 - 1 = 0$$

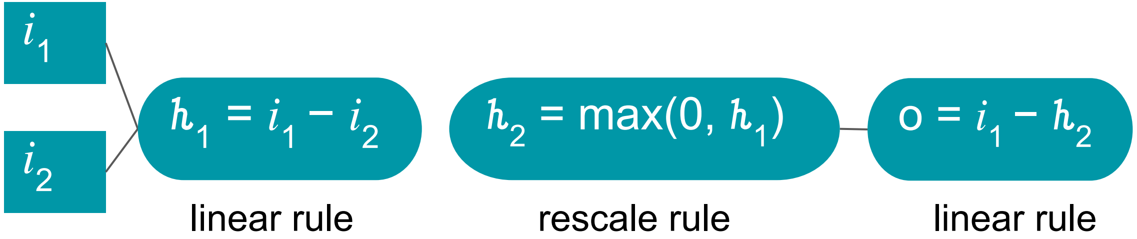
Chain rule : $m_{\Delta i_1 \Delta h_2} = m_{\Delta i_1 \Delta h_1} \cdot m_{\Delta h_1 \Delta h_2} = 1 \cdot 1 = 1$

$$m_{\Delta i_2 \Delta o} = -m_{\Delta i_2 \Delta h_2} = -1$$

Chain rule : $m_{\Delta i_2 \Delta h_2} = m_{\Delta i_2 \Delta h_1} \cdot m_{\Delta h_1 \Delta h_2} = -1 \cdot 1 = -1$

Example 2

$o = min(i_1, i_2)$



Thus, the contribution of i_1 and i_2 are given by

$$C_{\Delta i_1 \Delta o} = m_{\Delta i_1 \Delta o} \cdot \Delta i_1 = 0 \cdot i_1 = 0$$
$$C_{\Delta i_2 \Delta o} = m_{\Delta i_2 \Delta o} \cdot \Delta i_2 = -i_2$$

RevealCancel rule

Some functions may require treating the negative and positive values separately. Let $y = f(x)$ a non-linear function, then

$$\Delta y^+ = \frac{1}{2}\left(f(x^0+\Delta x^+) - f(x^0)\right) + \frac{1}{2}\left(f(x^0+\Delta x^- + \Delta x^+) - f(x^0+\Delta x^-)\right)$$

$$\Delta y^- = \frac{1}{2}\left(f(x^0+\Delta x^-) - f(x^0)\right) + \frac{1}{2}\left(f(x^0+\Delta x^+ + \Delta x^-) - f(x^0+\Delta x^+)\right)$$

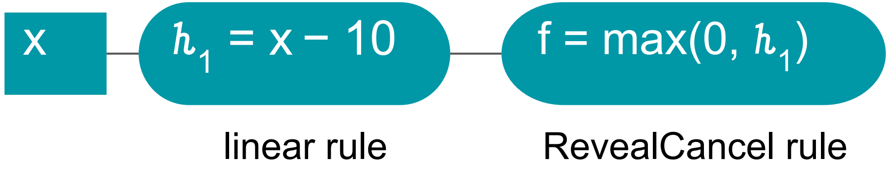
$$m_{\Delta x^+ \Delta y^+} = \frac{C_{\Delta x^+ y^+}}{\Delta x^+} = \frac{\Delta y^+}{\Delta x^+}\,;\, m_{\Delta x^- \Delta y^-} = \frac{\Delta y^-}{\Delta x^-}$$

It can be though as the Shapley values of Δx^+ and Δx^- contributing to y

The RevealCancel rule can replace the rescale rule, however in some cases it is preferable to use the rescale rule. Let us consider these cases

Example 1

$$f(x) = max(0, x - 10)$$



0. Initialisation

Let a reference point be $x = 0$ then $h_1 = -10$ and $f = 0$

Consider a point be $x = \varepsilon + 10$ then $h_1 = \varepsilon$ and $f = \varepsilon$

1. Linear rule

$m_{\Delta x^+ \Delta h_1^+} = 1$ (note that $m_{\Delta x^- \Delta h_1^+}$, $m_{\Delta x^- \Delta h_1^-}$, and $m_{\Delta x^+ \Delta h_1^-}$ do not exist, only one component is possible)

$$m_{\Delta x^+ \Delta h_1^+} \cdot \Delta x = C_{\Delta x^+ \Delta h_1^+} = 1 \cdot (\varepsilon + 10) = \varepsilon = \varepsilon + 10$$

Example 1

$$f(x) = max(0, x - 10)$$

2. Reveal Cancel Rule

$$\Delta f^{+} = \frac{1}{2}\Big[max(0, h_1^{0} + \Delta h_1^{+}) - max(0, h_1^{0}) + max(0, h_1^{0} + \Delta h_1^{+} + \Delta h_1^{-}) - max(0, h_1^{0} + \Delta h_1^{-}) \Big] = \left\{ \begin{array}{ll} h_1^{0} = h_1(x=0) = -10 & \\ \Delta h_1^{-} = 0 & \Delta h_1^{+} = \varepsilon - (-10) \end{array} \right\}$$

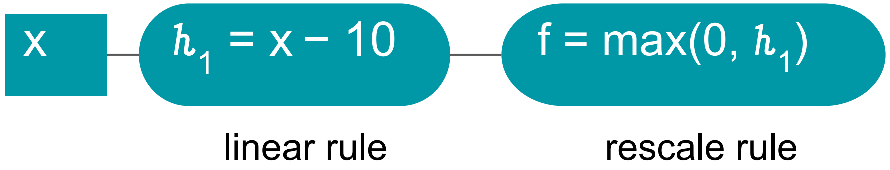
$$= \frac{1}{2}\Big[max(0, -10 + \varepsilon + 10) - 0 + max(0, -10 + \varepsilon + 10) - max(0, 0) \Big] =$$

$$= \frac{1}{2} \cdot 2 \cdot (\varepsilon) = \varepsilon$$

$$m_{\Delta h_1^{+} \Delta f^{+}} = \frac{\Delta f^{+}}{\Delta h_1^{+}} = \frac{\varepsilon}{\varepsilon - (-10)} = \frac{\varepsilon}{\varepsilon + 10}$$

Example 1

$$f(x) = max(0, x - 10)$$



3. Chain rule

$$m_{\Delta x \Delta f} = m_{\Delta x \Delta h_1} \cdot m_{\Delta h_1 \Delta f}$$

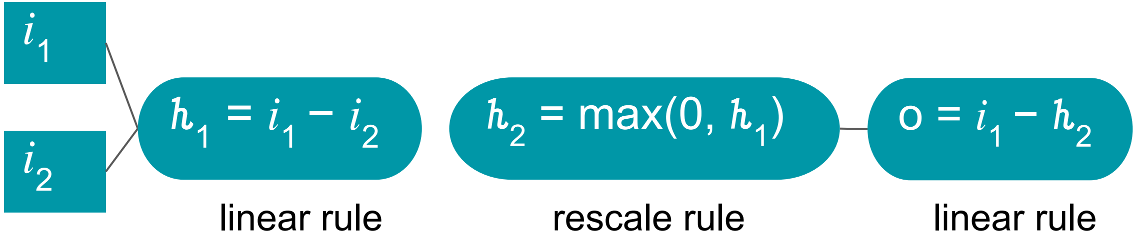
$$m_{\Delta x \Delta f} = 1 \cdot \frac{\varepsilon}{\varepsilon + 10}$$

4. Contribution

$$C_{\Delta x \Delta f} = m_{\Delta x \Delta f} \cdot \Delta x = \frac{\varepsilon}{\varepsilon + 10} \cdot (\varepsilon + 10) = \varepsilon$$

Example 2

$$o = min(i_1, i_2)$$



0. Initialisation

Let a reference point be $i_1 = i_2 = 0$ then $h_1 = 0$, $h_2 = 0$, $o = 0$

Consider points $i_1 > i_2 > 0$ then $h_1 = i_1 - i_2$, $h_2 = i_1 - i_2$, $o = i_1 - (i_1 - i_2) = i_2$

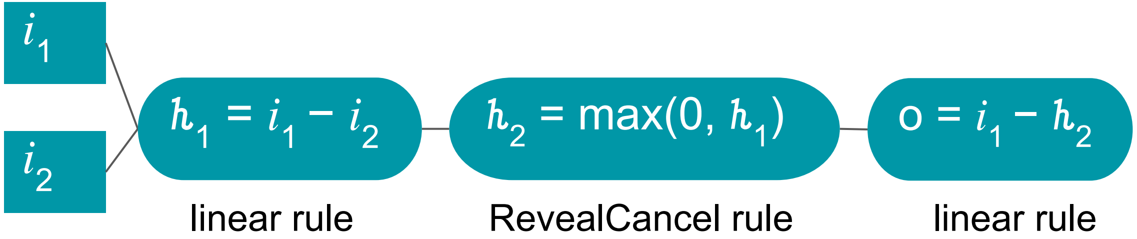
1. Linear rule

$$m_{\Delta i_1^+ \Delta h_1^+} = 1 \qquad C_{\Delta i_1^+ \Delta h_1^+} = i_1 \Rightarrow \Delta h_1^+ = i_1$$

$$m_{\Delta i_2^+ \Delta h_1^-} = -1 \quad C_{\Delta i_2^+ \Delta h_1^-} = -i_2 \Rightarrow \Delta h_1^- = -i_2$$

Example 2

$o = min(i_1, i_2)$



2. RevealCancel rule

$$\Delta h_2^+ = \frac{1}{2}[\max(0, h_1^0 + \Delta h_1^- + \Delta h_1^+) - \max(0, h_1^0 + \Delta h_1^-) + \max(0, h_1^0 + \Delta h_1^+) - \max(0, h_1^0)] =$$

$$\{\ h_1^0 = 0;\ \Delta h_1^+ = i_1;\ \Delta h_1^- = -i_2\ \}$$

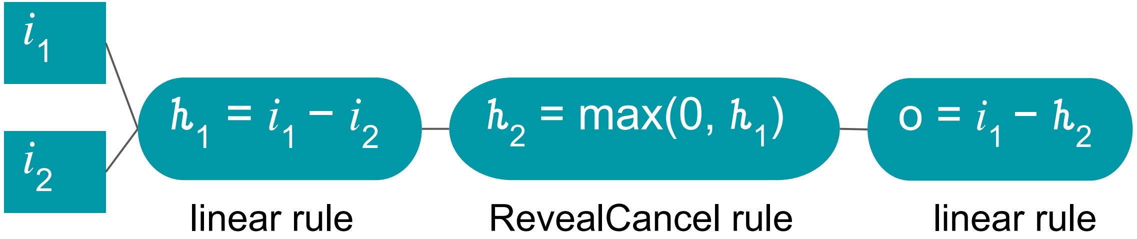
$$= \frac{1}{2}[\max(0, i_1 - i_2) - \max(0, -i_2) + \max(0, i_1) - \max(0, 0)] = \frac{1}{2}[(i_1 - i_2) + i_1] = \frac{1}{2}[2i_1 - i_2]$$

$$\Delta h_2^- = \frac{1}{2}[\max(0, h_1^0 + \Delta h_1^- + \Delta h_1^+) - \max(0, h_1^0 + \Delta h_1^+) + \max(0, h_1^0 + \Delta h_1^-) - \max(0, h_1^0)] =$$

$$= \frac{1}{2}[(i_1 - i_2) - i_1] = \frac{1}{2}[-i_2]$$

Example 2

$$o = min(i_1, i_2)$$



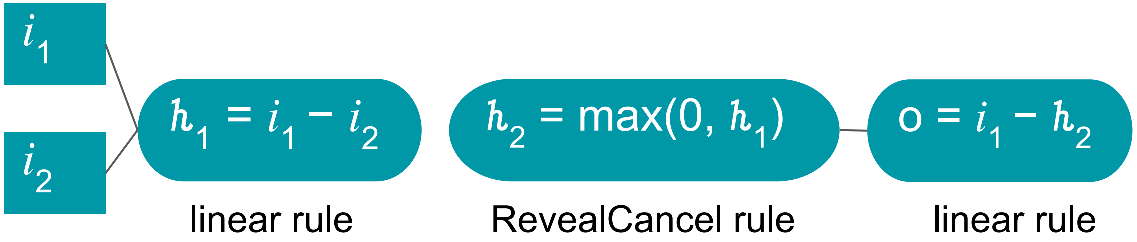
2. RevealCancel rule (continuation)

$$m_{\Delta h_1^+ \Delta h_2^+} = \frac{\frac{1}{2}(2i_1 - i_2)}{i_1} = \frac{i_1 - \frac{1}{2}i_2}{i_1}$$

$$m_{\Delta h_1^- \Delta h_2^-} = \frac{\frac{1}{2}(-i_2)}{-i_2} = \frac{1}{2}$$

Example 2

$$o = min(i_1, i_2)$$



3. Linear rule $$m_{\Delta i_1 \Delta h_2} = m_{\Delta i_1 \Delta h_1} \cdot m_{\Delta h_1^+ \Delta h_2^+} = 1 - \frac{1}{2} \cdot \frac{i_2}{i_1}$$

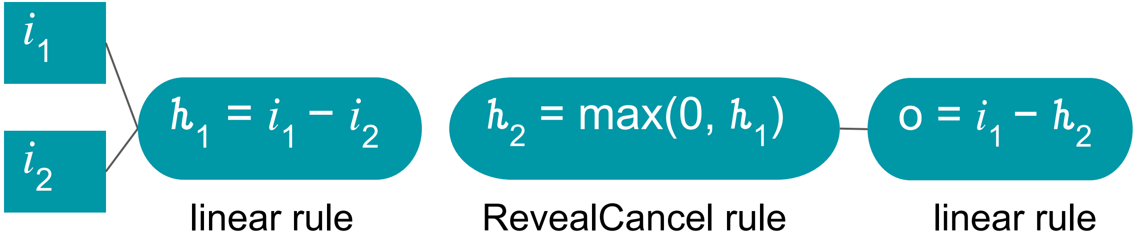
4. Chain rule $$m_{\Delta h_2 \Delta o} = -1$$

5. Linear rule $$m_{\Delta i_1 \Delta o} = 1 - m_{\Delta i_1 \Delta h_2} = 1 - m_{\Delta i_1 \Delta h_2} = 1 - \left(1 - \frac{1}{2} \cdot \frac{i_2}{i_1}\right) = \frac{1}{2} \cdot \frac{i_2}{i_1}$$

$$m_{\Delta i_2 \Delta o} = m_{\Delta i_2 \Delta h_1} \cdot m_{\Delta h_1^- \Delta h_2^-} \cdot m_{\Delta h_2 \Delta o} = -1 \cdot \left(\frac{1}{2}\right) \cdot (-1) = \frac{1}{2}$$

Example 2

$o = min(i_1, i_2)$



Thus, the contribution of i_1 and i_2 are given by

$$C_{\Delta i_1 \Delta o} = m_{\Delta i_1 \Delta o} \cdot \Delta i_1 = 0.5 \frac{i_2}{i_1} i_1 = 0.5 i_2$$

$$C_{\Delta i_2 \Delta o} = m_{\Delta i_2 \Delta o} \cdot \Delta i_2 = 0.5 i_2$$

When $i_1 < i_2$ then we have contribution $0.5 i_1$ for both inputs

Adjustment for the softmax layers

Motivation example: consider a sigmoid output o = σ(y), where y is the logit of the sigmoid function. Assume y = $x_1 + x_2$, where $x_{01} = x_{02} = 0$.

When $x_1 = 50$ and $x_2 = 0$, the output o saturates at very close to 1 and the contributions of x_1 and x_2 are 0.5 and 0 respectively

When $x_1 = 100$ and $x_2 = 100$, the output o is still very close to 1, but the contributions of x_1 and x_2 are now both 0.25.

This can be misleading when comparing scores across different inputs because a stronger contribution to the logit would not always translate into a higher DeepLIFT score.

Adjustment for the softmax layers

The final softmax output **involves a normalization over all classes**, but the linear layer before the softmax does not.

To address this, we can normalize the contributions to the linear layer by subtracting the mean contribution to all classes. Formally, if n is the number of classes, $C_{\Delta x \Delta c_i}$ represents the unnormalized contribution to class c_i in the linear layer and $C'_{\Delta x \Delta c_i}$ represents the normalized contribution, we have

$$C'_{\Delta x \Delta c_i} = C_{\Delta x \Delta c_i} - \frac{1}{n} \sum_{j=1}^{n} C_{\Delta x \Delta c_j}$$

As a justification for this normalization, we note that subtracting a fixed value from all the inputs to the softmax leaves the output of the softmax unchanged

Wrapping up...

Benefits:

- Dealing with the **saturation** and **threshold** problems
- Providing a more **reasonable explanation** for some complex functions

Questions to study:

- Application to RNNs
- Computing a good reference
- Defining better rules (beyond gradients) for max-neurons (maxout/maxpooling)

Integrated Gradients

Motivation: methods are unsupervised, thus it is hard to evaluate them empirically

Solution: to use axioms (i.e., desired properties) and evaluate methods w.r.t. them

Basic settings: for each image I we also have a ***baseline*** (reference) image

If we search a cause of something on I then the baseline image is the image that does not contain the cause, e.g., black image for the object recognition task or the zero embedding vector for text models

Sensitivity axiom

Axiom: a method satisfies **sensitivity** if for every input and baseline that differ in one feature but have different predictions then the differing feature should be given a non-zero attribution

Example: $f(x) = 1 - \mathrm{ReLU}(1 - x)$, the baseline is $x = 0$, the input is $x = 2$

$x = 0$ $f(0) = \mathbf{0}$

$x = 2$ $f(1) = \mathbf{1}$

$df/dx = 1$ if $x < 1$, else 0

Thus, for $x = 2$ `Gradient` and `Input * Gradient` give importance 0

Sensitivity axiom

methods VIOLATING the axiom

Gradients

DeconvNet

Guided backpropagation

methods SUPPORTING the axiom

LRP

DeepTaylor

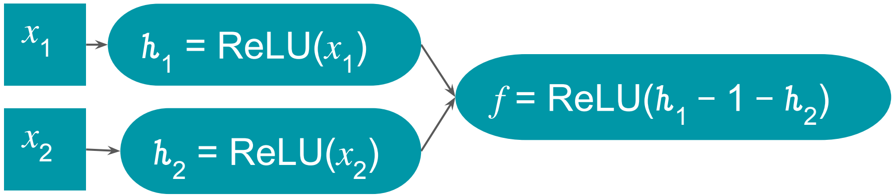
DeepLIFT

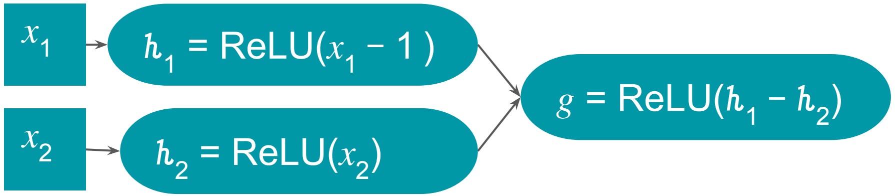
Implementation invariance axiom

Two networks are **functionally equivalent** if their outputs are equal for all inputs, despite having very different implementations

Axiom: the attributions are always identical for two functionally equivalent networks

Example: consider $x_1 = 3$, $x_2 = 1$ and the following two networks and the reference point $x_1 = 0$, $x_2 = 0$



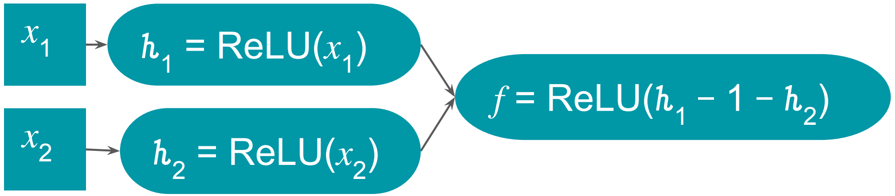


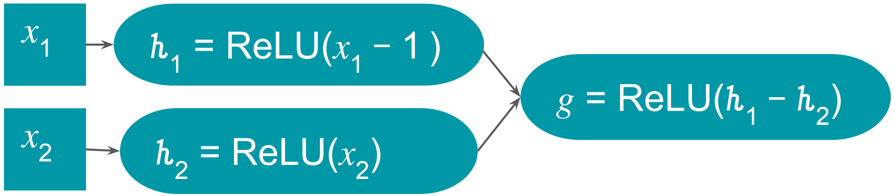
Implementation invariance axiom

methods SUPPORTING the axiom	**methods VIOLATING the axiom**
Gradients	DeepLIFT
DeconvNet	DeepTaylor
Guided backpropagation	LRP

Example of the DeepLIFT application

Example: at $x_1 = 3$, $x_2 = 1$ with the reference point $x_1 = 0$, $x_2 = 0$





Attributions:

Integrated gradients $x_1 = 1.5$, $x_2 = -0.5$

DeepLIFT $x_1 = 1.5$, $x_2 = -0.5$

LRP $x_1 = 1.5$, $x_2 = -0.5$

Attributions:

Integrated gradients $x_1 = 1.5$, $x_2 = -0.5$

DeepLIFT $x_1 = 2$, $x_2 = -1$

LRP $x_1 = 2$, $x_2 = -1$

Sundararajan, Mukund, Ankur Taly, and Qiqi Yan. "Axiomatic attribution for deep networks." International Conference on Machine Learning. PMLR, 2017

Integrated gradients. Main steps

Step 1: Find a baseline model

Step 2: Generate a linear interpolation between the baseline and the original image

Step 3: Calculate gradients (to get pixels with the strongest effects on the output) to measure the relationship between changes to a feature and changes in the model's predictions

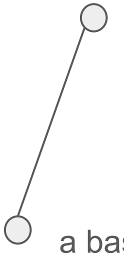
Step 4: Compute the numerical approximation through averaging gradients

Step 5: Scale IG to the input image

Integrated Gradients

Let f: $\mathbb{R}^n \rightarrow [0,1]$ be a function that represents a deep network

Idea: to consider a straight line connecting two points and to compute the gradients at each point of this path and then commutaling them all



The integrated gradient along the i-th dimension for x and x' is defined as follows:

$$IG_i(x) = (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial f(x' + \alpha(x - x'))}{\partial x_i} d\alpha$$

Approximated gradients

$$IG_i^{approx}(x) = (x_i - x_i') \times \sum_{k=1}^{m} \frac{\partial F(x' + \frac{k}{m}(x - x'))}{\partial x_i} \frac{1}{m}$$

Completeness axiom

The attributions add up to the difference between the output of f at x and x'

$$\sum_{i=1}^{n} IG_i(x) = f(x) - f(x')$$

Completeness axiom → sensitivity axiom

Remark: this axiom is desirable in LRP and DeepLIFT

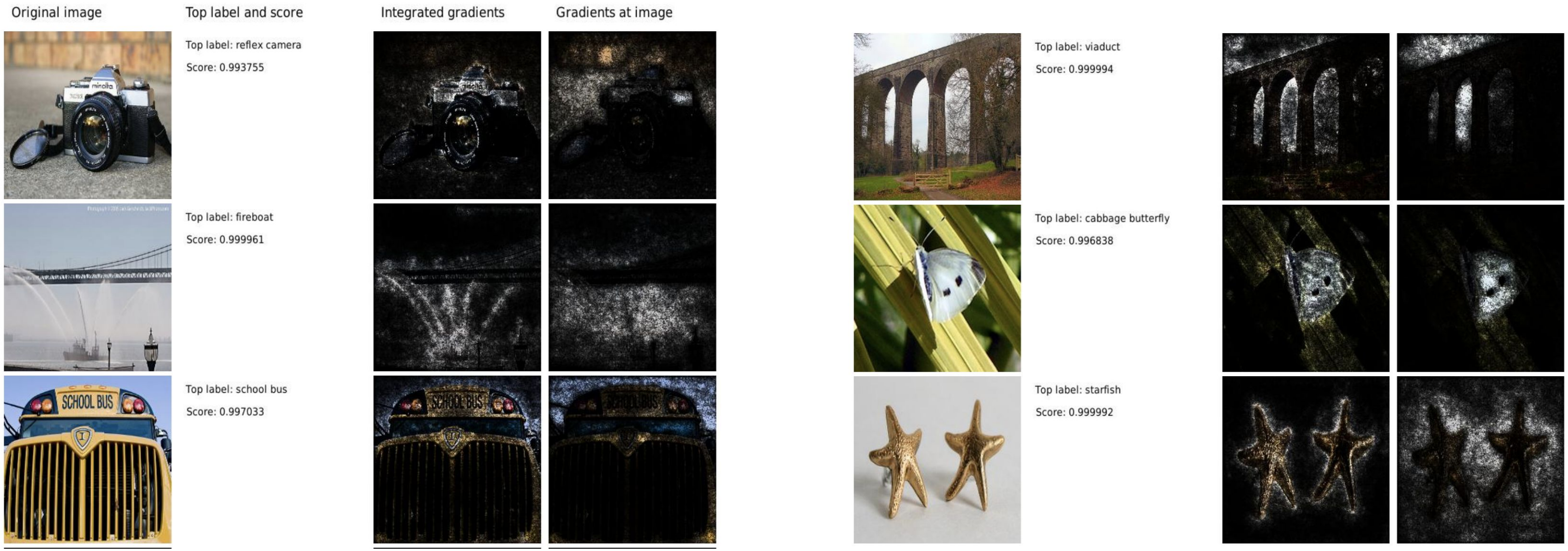
Dummy sensitivity axiom

If the function implemented by the deep network does not depend (mathematically) on some variable, then the attribution to that variable is always zero

Linearity axiom

Suppose that we linearly composed two deep networks modeled by the functions f_1 and f_2 to form a third network that models the function $a \cdot f_1 + b \cdot f_2$, i.e., a linear combination of the two networks. Then we'd like the attributions for $a \cdot f_1 + b \cdot f_2$ to be the weighted sum of the attributions for f_1 and f_2 with weights a and b respectively

Experiments



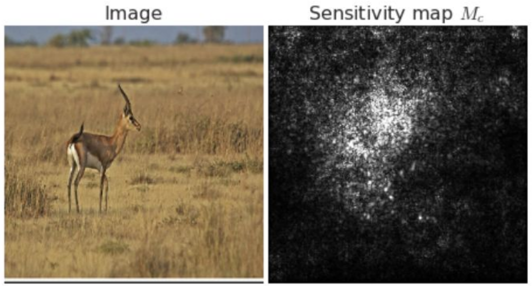
Useful links

- DeepLIFT package (for TF1): https://github.com/kundajelab/deeplift
- Tutorial on Integrated gradients https://www.tensorflow.org/tutorials/interpretability/integrated_gradients

SmoothGrad

Motivation: gradient-based approaches are sensitive to noise, the sensitivity map are not sharp enough

Idea: **reduce visual noise** for a given image by **sampling** similar images (by adding noise), and then **averaging** the resulting sensitivity maps for each sampled image



$$class(x) = \operatorname{argmax}_{c \in C} S_c(x)$$

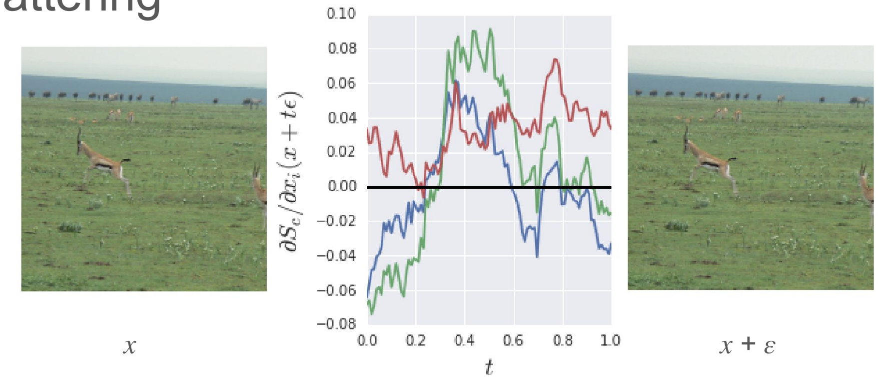
$$M_c(x) = \partial S_c(x) / \partial x$$

Remark: training with noise and inferring with noise may provide even better results than just inferring with noise

Smilkov, D., Thorat, N., Kim, B., Viégas, F., & Wattenberg, M. (2017). Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*.

Issues related to back-propagation

- Saturation
- Strong influence of distractors (see PatternNet/PatternAttribution)
- Gradient shattering
- ...



The partial derivative of Sc with respect to the RGB values of a single pixel as a fraction of the maximum entry in the gradient vector $\max_i \partial Sc/\partial x_i(t)$, (middle plot) as one slowly moves away from a baseline image x (left plot) to a fixed location $x + \varepsilon$ (right plot)

ε is one random sample from $N(0, 0.012)$

SmoothGrad

We can smooth the gradients with a Gaussian kernel, however it is too expensive.

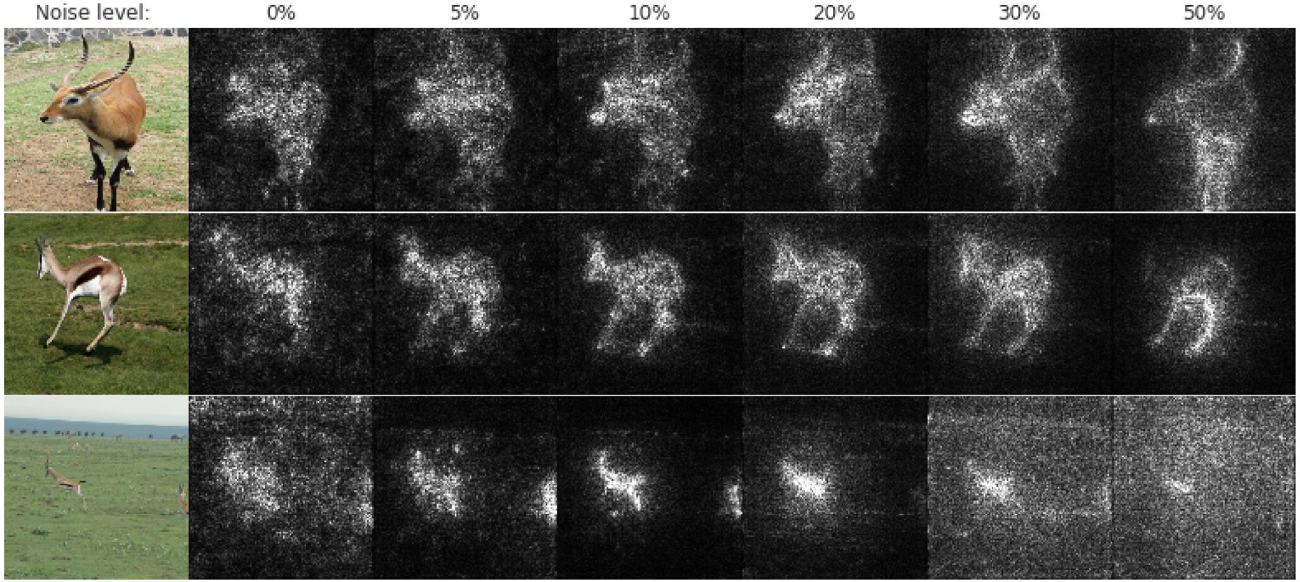
Thus, one uses its approximation

$$\hat{M}_c(x) = \frac{1}{n}\sum_{1}^{n} M_c(x + \mathcal{N}(0, \sigma^2))$$

where n is the number of samples, and $N(0, \sigma^2)$ represents Gaussian noise with standard deviation σ

Parameters to set: n and σ

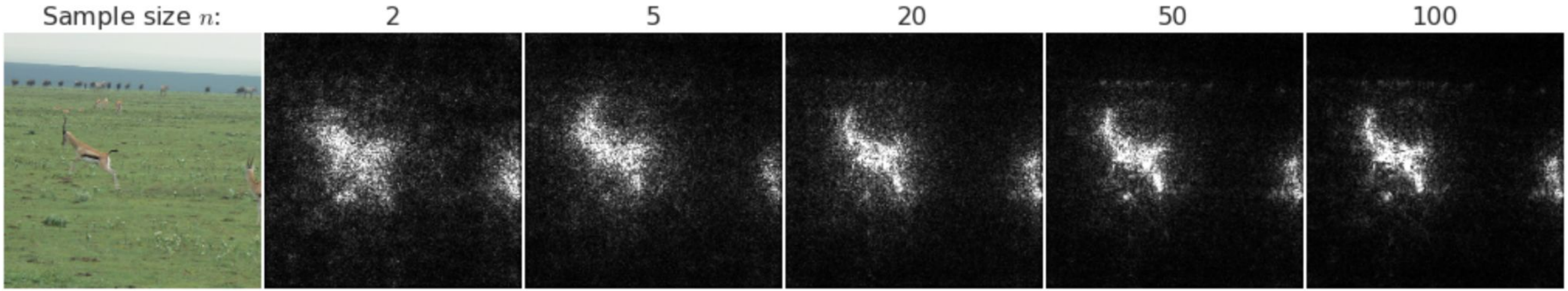
Choosing an optimal noise rate for SmoothGrad



Effect of noise level (columns) on our method for 5 images of the gazelle class in ImageNet (rows). Each sensitivity map is obtained by applying Gaussian noise $N(0, \sigma^2)$ to the input pixels for 50 samples, and averaging them. The noise level corresponds to $\sigma/(x_{max} - x_{min})$

Conclusion: applying 10% - 20% noise (middle columns) seems to balance the sharpness of sensitivity map and maintain the structure of the original image, however the ideal noise level may depend on the input

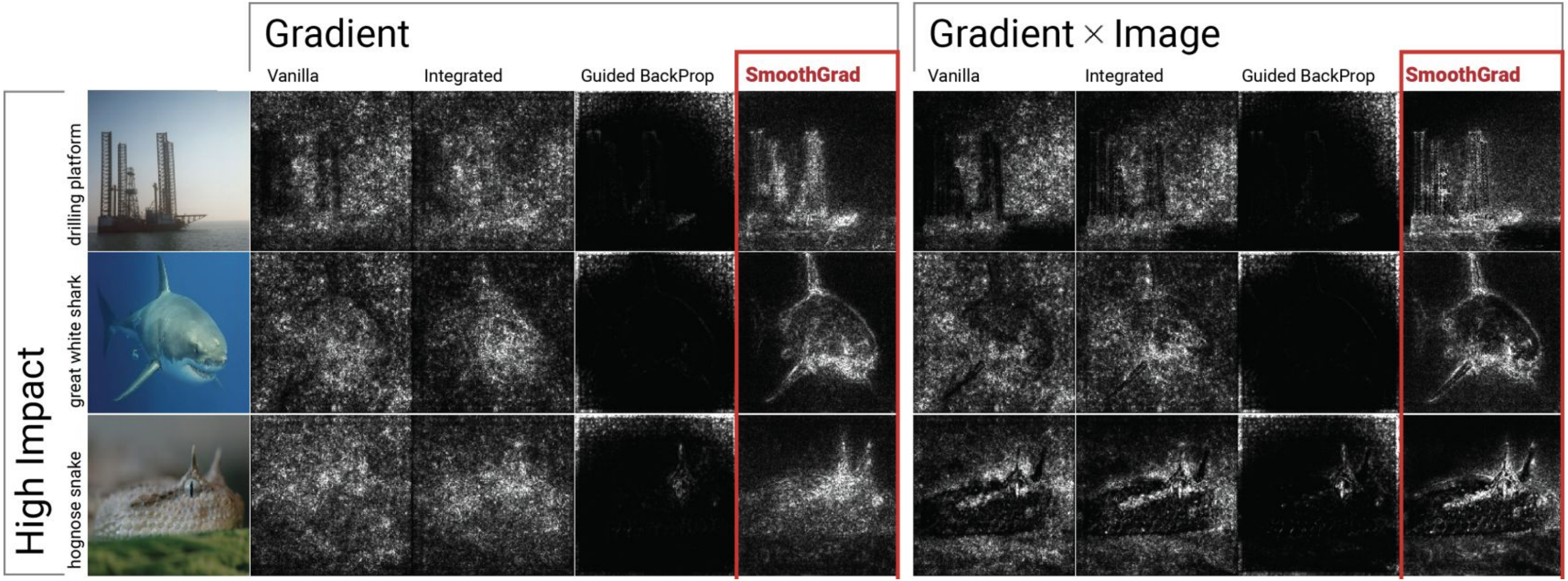
Choosing an optimal sample size for SmoothGrad



Effect of sample size on the estimated gradient for Inception v3 model. 10% noise was applied to each image

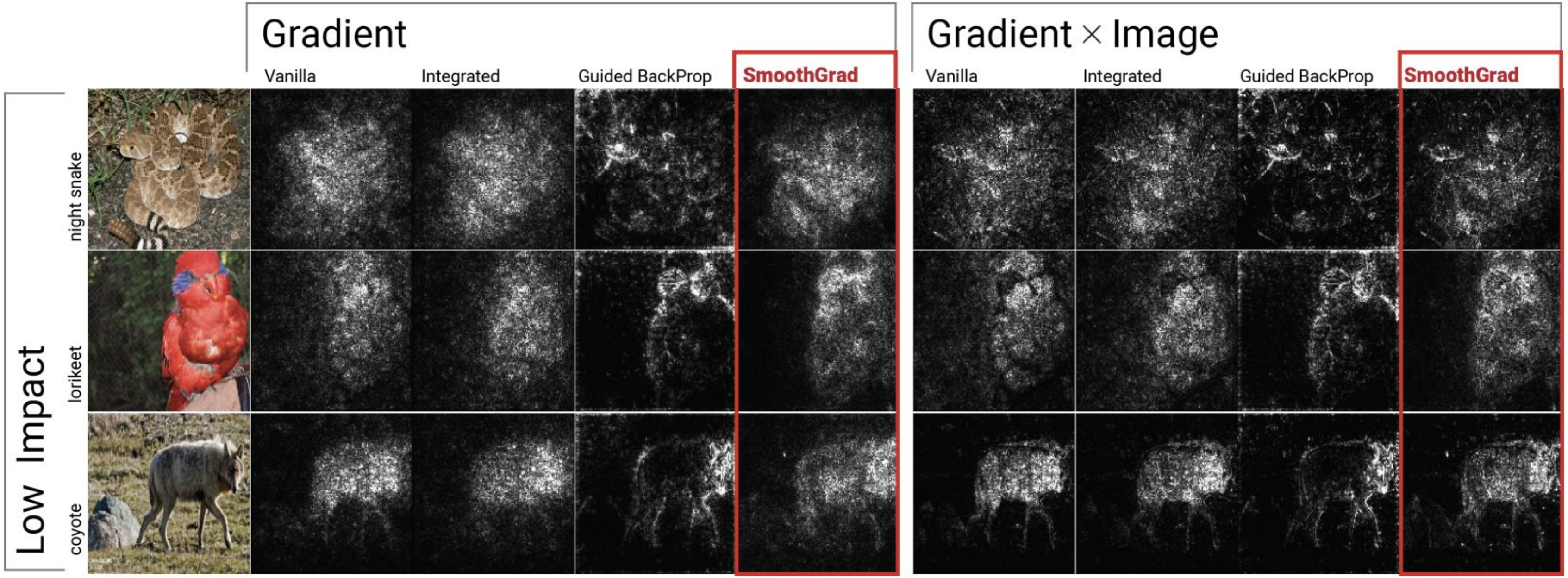
Conclusion: the estimated gradient becomes smoother as the sample size, increases. There is little apparent change in the visualizations for n > 50.

Application



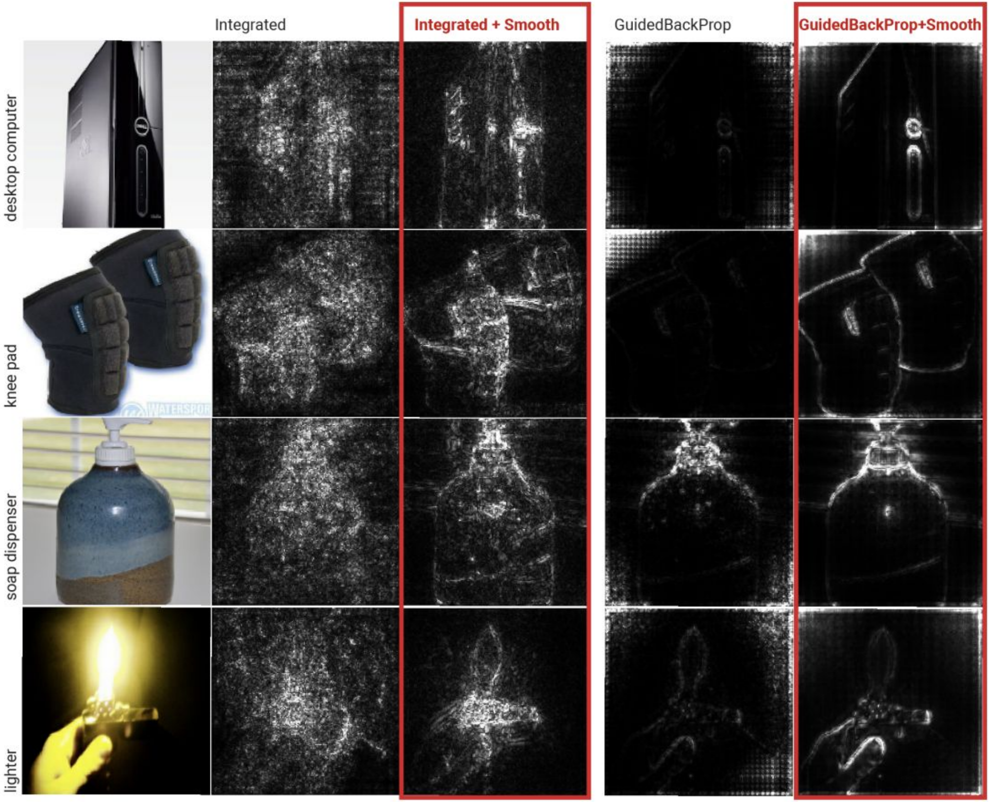
SmoothGrad is applied to Vanilla Gradient

Application



SmoothGrad is applied to Vanilla Gradient

Combining SmoothGrad and other methods

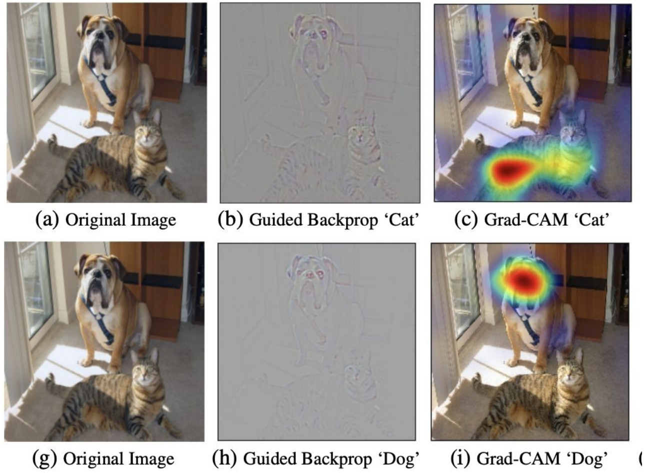


Grad-CAM (Class Activation Mapping)

Motivation: good visual explanation should be

(a) high-resolution, i.e. capture fine-grained detail ok for all aforementioned methods

(b) class discriminative, i.e. localize the category in the image, not a case for aforementioned methods



GradCAM

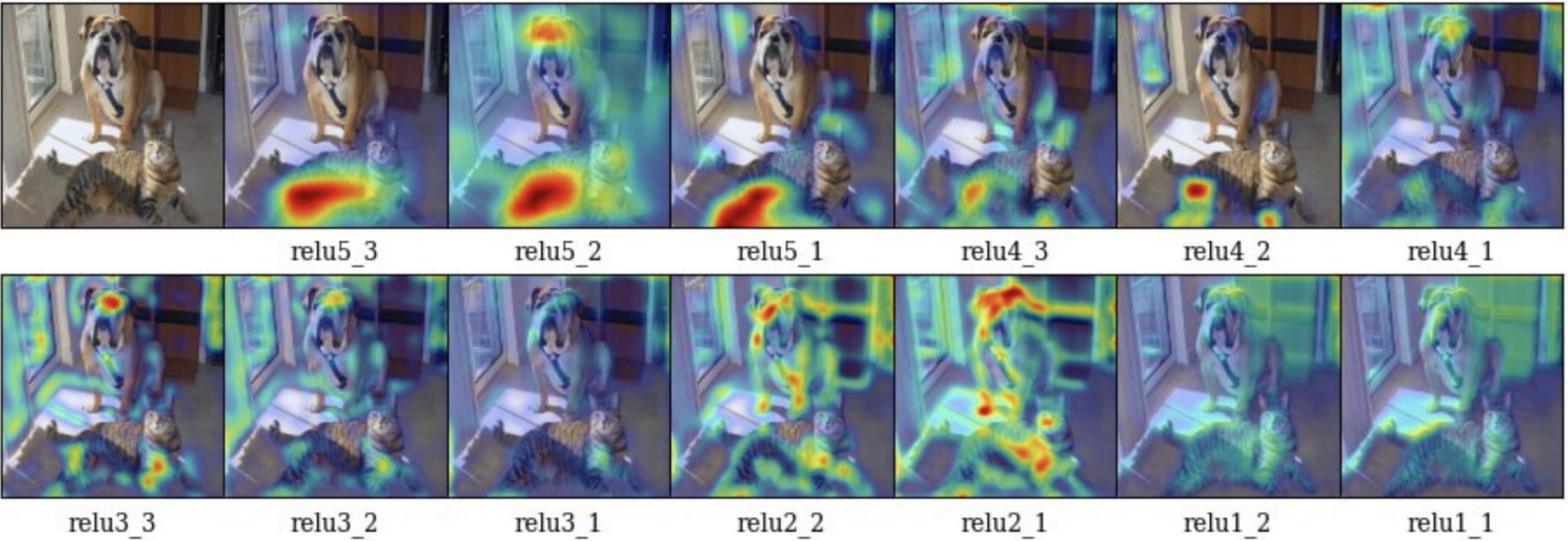
Importance scores of the k-th feature map in the last convolutional layer for class c are given by

$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i \sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}}$$

Then each activation map is multiplied by its importance and only positive scores are taken into account:

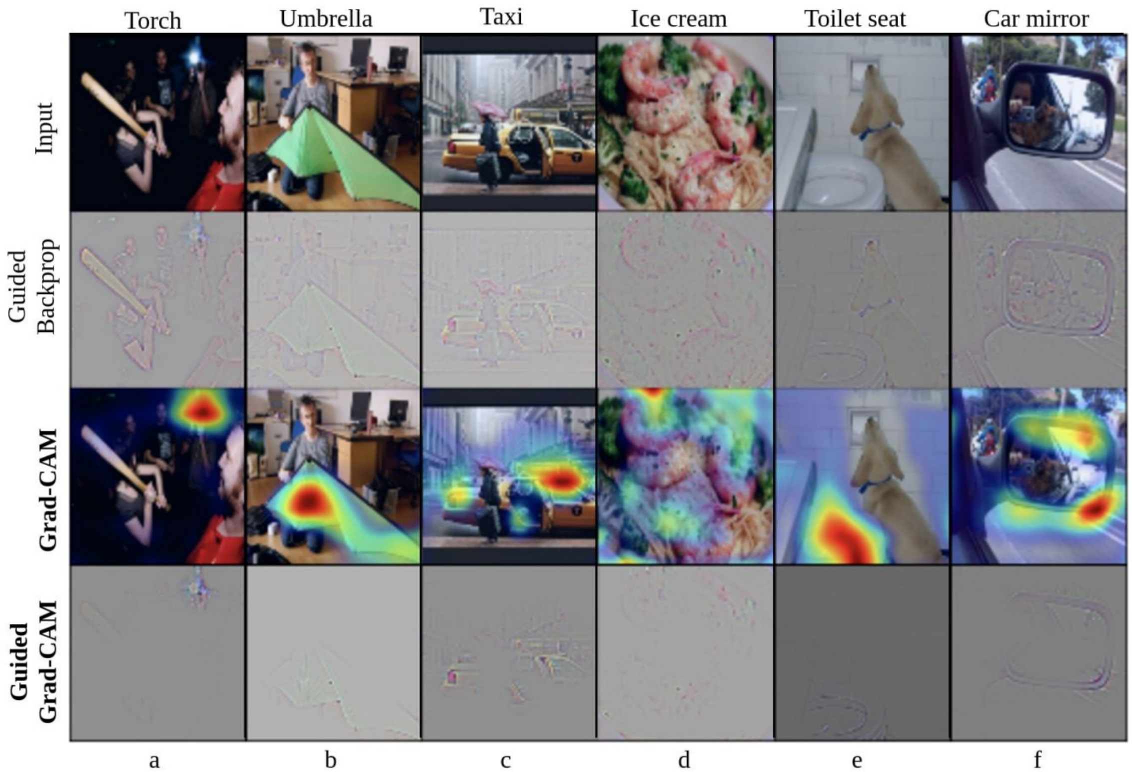
$$L_{\text{Grad-CAM}}^c = ReLU \underbrace{\left(\sum_k \alpha_k^c A^k \right)}_{\text{linear combination}}$$

Visualization of different feature maps



Grad-CAM at different convolutional layers for the ‘tiger cat’ class. This figure analyzes how localizations change qualitatively as we perform Grad-CAM with respect to different feature maps in a CNN (VGG16 [52]). We find that the best looking visualizations are often obtained after the deepest convolutional layer in the network, and localizations get progressively worse at shallower layers. This is consistent with our intuition described in Section 3 of main paper, that deeper convolutional layer capture more semantic concepts

Guided Backpropagation × GradCAM



Visualizations for randomly sampled images from the COCO validation dataset. Predicted classes are mentioned at the top of each column