Axiomatic Attribution for deep networks

Mukund Sundararajan, Ankur Taly, Qiqi Yan

Attributing the prediction of a deep network to its input features (a problem previously studied by several other works)

Abstract -- attributing the prediction of a deep network to its input features

- They identify two fundamental axioms: Sensitivity + Implementation Invariance
- They show that the two not satisfied by most known attribution methods (a fundamental weakness of those methods).
- Integrated Gradients -- a new attribution method:
  - requires no modification to the original network
  - Extremely simple to implement
- apply this method to a couple of image models, a couple of text models and a chemistry model

Instance for (definition 1?) attribution method

For instance, in an object recognition network, an attribution method could tell us which pixels of the image were responsible for a certain label being picked (see Figure 2). The attribution problem was previously studied by various papers (Baehrens et al., 2010; Simonyan et al., 2013; Shrikumar et al., 2016; Binder et al., 2016; Springenberg et al., 2014).

Definition [edit source]

The score is the gradient (the vector of partial derivatives), with respect to some parameter θ, of the logarithm (commonly the natural logarithm) of the likelihood function (the log-likelihood). If the observation is X and its likelihood is $\mathcal{L}(\theta; X)$, then the score V can be found through the chain rule:

$$V \equiv V(\theta, X) = \frac{\partial}{\partial \theta} \log \mathcal{L}(\theta; X) = \frac{1}{\mathcal{L}(\theta; X)} \frac{\partial \mathcal{L}(\theta; X)}{\partial \theta}.$$

Thus the score V indicates the sensitivity of $\mathcal{L}(\theta; X)$ (its derivative normalized by its value). Note that V is a function of θ and the observation X, so that, in general,

Scores (in figure 2) is talking about sensitivity ? w.r.t. parameters according to https://en.wikipedia.org/wiki/Score_(statistics)#Definition

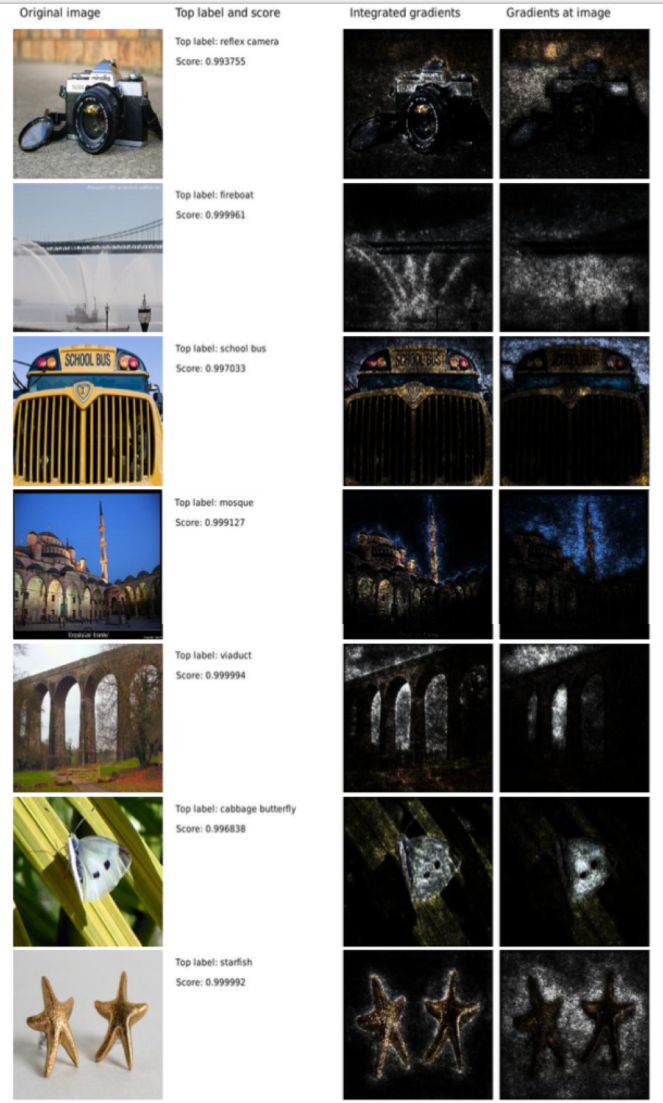

Figure 2. **Comparing integrated gradients with gradients at the image.** Left-to-right: original input image, label and softmax score for the highest scoring class, visualization of integrated gradients, visualization of gradients*image. Notice that the visualizations obtained from integrated gradients are better at reflecting distinctive features of the image.

More words for attribution method

- The intention of these works is to understand the input-output behavior of the deep network, which gives us the ability to improve it. Such understandability is critical to all computer programs, including machine learning models. There are also other applications of attribution. They could be used within a product driven by machine learning to provide a rationale for the recommendation.
- For instance, a deep network that predicts a condition based on imaging could help inform the doctor of the part of the image that resulted in the recommendation. This could help the doctor understand the strengths and weaknesses of a model and compensate for it. We give such an example in Section 6.2. Attributions could also be used by developers in an exploratory sense.
- For instance, we could use a deep network to extract insights that could be then used in a rule-based system. In Section6.3, we give such an example.

A significant challenge in designing an attribution technique

A significant challenge in designing an attribution technique is that they are hard to evaluate empirically. As we discuss in Section 4, it is hard to tease apart errors that stem from the misbehavior of the model versus the misbehavior of the attribution method. To compensate for this shortcoming, we take an axiomatic approach. In Section 2 we identify two axioms that every attribution method must satisfy. Unfortunately most previous methods do not satisfy one of these two axioms. In Section 3, we use the axioms to identify a new method, called **integrated gradients**.

Before two axioms definition 1

Probability for certain label

- Axioms will mention concept like “baseline”

We study the problem of *attributing the prediction of a deep network to its input features.*

Definition 1. *Formally, suppose we have a function* $F : \mathsf{R}^n \rightarrow [0, 1]$ *that represents a deep network, and an input* $x = (x_1, \ldots, x_n) \in \mathsf{R}^n$. *An attribution of the prediction at input* x *relative to a baseline input* x' *is a vector* $A_F(x, x') = (a_1, \ldots, a_n) \in \mathsf{R}^n$ *where* a_i *is the* contribution *of* x_i *to the prediction* $F(x)$.

The need for the baseline in definition 1

Remark 1. *Let us briefly examine the need for the baseline in the definition of the attribution problem. A common way for humans to perform attribution relies on counterfactual intuition. When we assign blame to a certain cause we implicitly consider the absence of the cause as a baseline for comparing outcomes. In a deep network, we model the absence using a single baseline input. For most deep networks, a natural baseline exists in the input space where the prediction is neutral. For instance, in object recognition networks, it is the black image. The need for a baseline has also been pointed out by prior work on attribution (Shrikumar et al., 2016; Binder et al., 2016).*

Control group
&
Experimental/
treatment group

Two fundmental axioms -- axiom 1 sensitivity

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

Gradients violates sensitivity (a)

- Gradients: For linear models, ML practitioners regularly inspect the products of the model coefficients? and the feature values in order to debug predictions. Gradients (of the output with respect to the input) is a natural analog of the model coefficients for a deep network, and therefore the product of the gradient and feature values is a reasonable starting point for an attribution method (Baehrens et al., 2010; Simonyan et al., 2013); see the third column of Figure 2 for examples. The problem with gradients is that they break *sensitivity*, a property that all attribution methods should satisfy.

Gradients violates sensitivity (a)

Gradients violate Sensitivity(a): For a concrete example, consider a one variable, one ReLU network, $f(x) = 1 - \text{ReLU}(1-x)$. Suppose the baseline is $x = 0$ and the input is $x = 2$. The function changes from 0 to 1, but because f becomes flat at $x = 1$, the gradient method gives attribution of 0 to x. Intuitively, gradients break Sensitivity because the prediction function may flatten at the input and thus have zero gradient despite the function value at the input being different from that at the baseline. This phenomenon has been reported in previous work (Shrikumar et al., 2016).

Maybe just HEURISTC: for example, you can analyze this case in a different way (since you are talking about baseline here):

why not compare two gradients? – then you can see differences!!

the paper Hengchu had not finished discussing.

Gradients violates sensitivity (a)

- "Gradient-based approaches are problematic because activation functions such as ReLUs have a gradient of zero when they are not firing, and yet a ReLU that does not fire can still carry information (Figure 1). Similarly, sigmoid or tanh activations are popular choices for the activation functions of gates …"

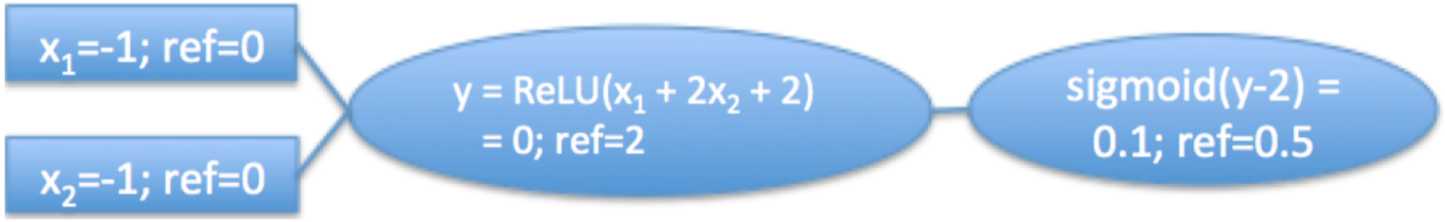

Figure 1. Simple network with inputs x_1 and x_2 that have reference values of 0. When $x_1 = x_2 = -1$, output is 0.1 but the gradients w.r.t x_1 and x_2 are 0 due to inactive ReLU y (which has activation of 2 under reference input). By comparing activations to their reference values, DeepLIFT assigns contributions to the output of $\left((0.1 - 0.5)\frac{1}{3}\right)$ to x_1 and $\left((0.1 - 0.5)\frac{2}{3}\right)$ to x_2.

Shrikumar, Avanti, et al. "Not just a black box: Learning important features through propagating activation differences." *arXiv preprint arXiv:1605.01713* (2016).

Two small questions: what is "firing", what are "reference input, reference value" for neural activations?

Gradients violates sensitivity (a)

- Practically, the lack of sensitivity causes gradients to focus on irrelevant features (see the “fireboat” example in Figure 2).

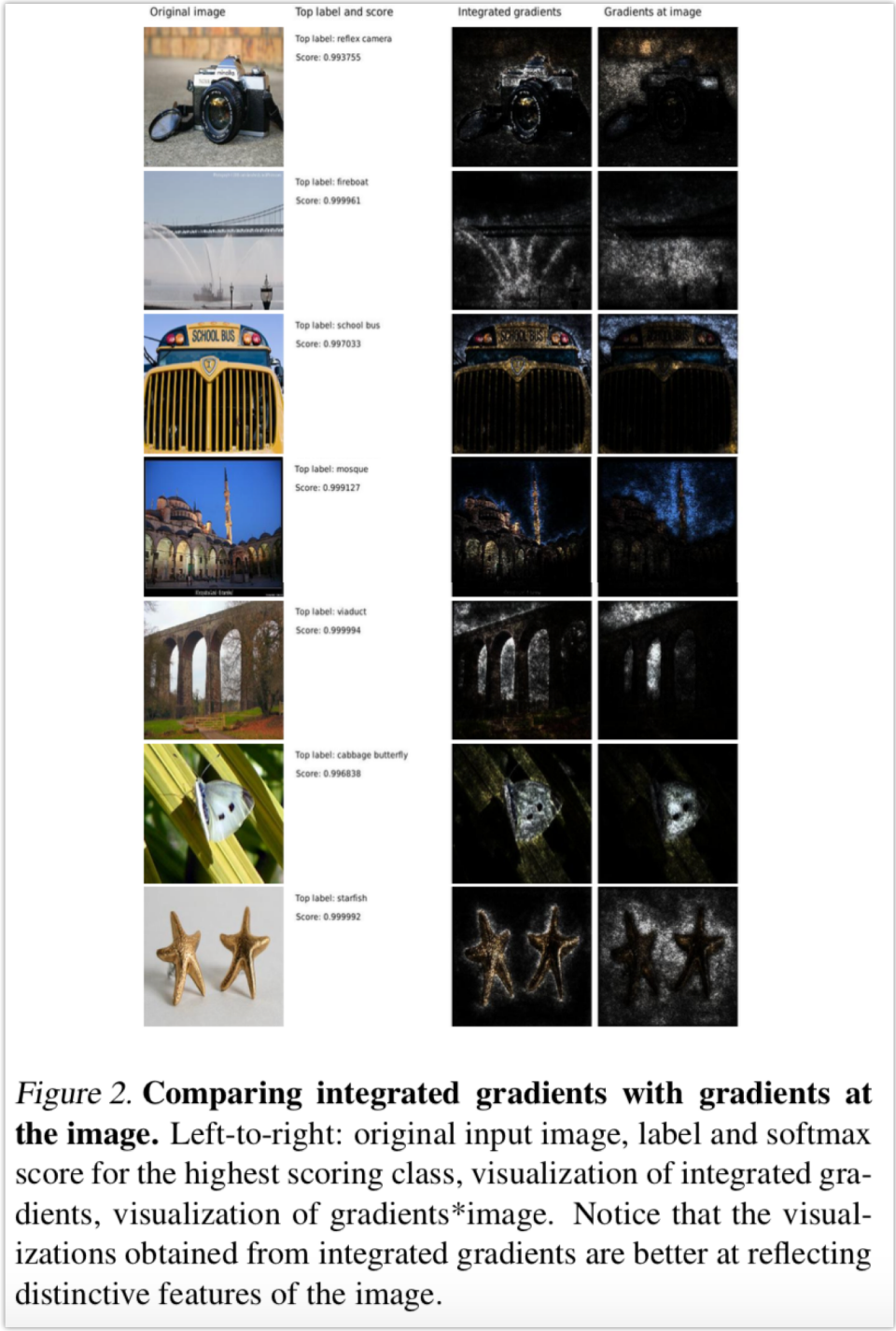

Figure 2. **Comparing integrated gradients with gradients at the image.** Left-to-right: original input image, label and softmax score for the highest scoring class, visualization of integrated gradients, visualization of gradients*image. Notice that the visualizations obtained from integrated gradients are better at reflecting distinctive features of the image.

Deconvolution networks (DeConvNets), and Guided back-propagation violate Sensitivity(a)

- **Other back-propagation based approaches:**
  A second set of approaches involve back-propagating the final prediction score through each layer of the network down to the individual features. These include DeepLift (Hengchu's again), Layer-wise relevance propagation (LRP), Deconvolutional networks (DeConvNets), and Guided back-propagation. These methods differ in the specific backpropagation logic for various activation functions (e.g., ReLU, MaxPool, etc.).

  How different?

Deconvolution networks (DeConvNets), and Guided back-propagation violate Sensitivity(a)

Unfortunately, Deconvolution networks (DeConvNets), and Guided back-propagation violate Sensitivity(a).
This is because these methods back-propogate through a ReLU node only if the ReLU is turned on at the input. This makes the method similar to gradients, in that, the attribution is zero for features with zero gradient at the input despite a non-zero gradient at the baseline. We defer the specific counterexamples to Appendix B.

How different? At this step, the essential is the same to me.

DeepLift & LRP violate a different requirement on attribution methods

- Methods like DeepLift and LRP tackle the Sensitivity issue by employing a baseline, and in some sense try to compute ``discrete gradients'' instead of (instantaeneous) gradients at the input. (The two methods differ in the specifics of how they compute the discrete gradient).
  But the idea is that a large, discrete step will avoid flat regions, avoiding a breakage of sensitivity. Unfortunately, these methods violate a different requirement on attribution methods.

  What is "a different requirement on attribution methods"? Next axiom: implementation invariance.

Axiom: implementation invariance

- Two networks are ***functionally equivalent*** if their outputs are equal for all inputs, despite having very different implementations. Attribution methods should satisfy ***Implementation Invariance***, i.e., the attributions are always identical for two functionally equivalent networks.
  Any practically nontrivial example of a pair of two functionally equivalent networks?

  To motivate this, notice that attribution can be colloquially defined as assigning the blame (or credit) for the output to the input features. Such a definition does not refer to implementation details.
  Colloquially/ heursically – might can be formulated better

DeepLift and LRP violate Axiom 2

First, notice that gradients are invariant to implementation. In fact, the chain-rule for gradients $\frac{\partial f}{\partial g} = \frac{\partial f}{\partial h} \cdot \frac{\partial h}{\partial g}$ is essentially about implementation invariance. To see this, think of g and f as the input and output of a system, and h being some implementation detail of the system. The gradient of output f to input g can be computed either directly by $\frac{\partial f}{\partial g}$, ignoring the intermediate function h (implementation detail), or by invoking the chain rule via h. This is exactly how backpropagation works.

DeepLift and LRP violate Axiom 2

Methods like LRP and DeepLift replace gradients with discrete gradients and still use a modified form of backpropagation to compose discrete gradients into attributions. Unfortunately, the chain rule does not hold for discrete gradients in general. Formally $\frac{f(x_1)-f(x_0)}{g(x_1)-g(x_0)} \neq \frac{f(x_1)-f(x_0)}{h(x_1)-h(x_0)} \cdot \frac{h(x_1)-h(x_0)}{g(x_1)-g(x_0)}$, and therefore these methods fail to satisfy implementation invariance.

Violation of Axiom 2

If an attribution method fails to satisfy Implementation Invariance, the attributions are potentially sensitive to unimportant aspects of the models. For instance, if the network architecture has more degrees of freedom than needed to represent a function then there may be two sets of values for the network parameters that lead to the same function. The training procedure can converge at either set of values depending on the initializtion or for other reasons, but the underlying network function would remain the same. It is undesirable that attributions differ for such reasons.

Just nonidentifiability or overfitting?

Our method: integrated gradients

- Intuitively, our techniue combines the implementation invariance of Gradients along with the Sensitivity of tehniques like LRP or DeepLift.

Formally, suppose we have a function $F : \mathrm{R}^n \to [0, 1]$ that represents a deep network. Specifically, let $x \in \mathrm{R}^n$ be the input at hand, and $x' \in \mathrm{R}^n$ be the baseline input. For image networks, the baseline could be the black image, while for text models it could be the zero embedding vector.

Our method: integrated gradients

We consider the straightline path (in R^n) from the baseline x' to the input x, and compute the gradients at all points along the path. Integrated gradients are obtained by cumulating these gradients. Specifically, integrated gradients are defined as the path intergral of the gradients along the straightline path from the baseline x' to the input x.

The integrated gradient along the i^{th} dimension for an input x and baseline x' is defined as follows. Here, $\frac{\partial F(x)}{\partial x_i}$ is the gradient of $F(x)$ along the i^{th} dimension.

$$\mathsf{IntegratedGrads}_i(x) ::= (x_i - x'_i) \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} \, d\alpha \tag{1}$$

Integrated gradients satisfies Axiom 3

Axiom: Completeness. Integrated gradients satisfy an axiom called *completeness* that the attributions add up to the difference between the output of F at the input x and the *baseline* x'.

In definition 1, we definitely have $A_F(x, x')$, but do we have F(x')?

This axiom is identified as being desirable by Deeplift and LRP. It is a sanity check that the attribution method is somewhat comprehensive in its accounting, a property that is clearly desirable if the networks score is used in a numeric sense, and not just to pick the top label, for e.g., a model estimating insurance premiums from credit features of individuals.

Integrated gradients satisfies Axiom 3

This is formalized by the proposition below, which instantiates the fundamental theorem of calculus for path integrals.

Proposition 1. *If* $F : \mathsf{R}^n \to \mathsf{R}$ *is differentiable almost everywhere* [1] *then*

$$\Sigma_{i=1}^{n} \mathsf{IntegratedGrads}_i(x) = F(x) - F(x')$$

For most deep networks, it is possible to choose a baseline such that the prediction at the baseline is near zero ($F(x') \approx 0$). (For image models, the black image baseline indeed satisfies this property.) In such cases, there is an intepretation of the resulting attributions that ignores the baseline and amounts to distributing the output to the individual input features.

Integrated gradients satisfies Axiom 3 & axiom 1(a)

Remark 2. *Integrated gradients satisfies Sensivity(a) because Completeness implies Sensivity(a) and is thus a strengthening of the Sensitivity(a) axiom. This is because Sensitivity(a) refers to a case where the baseline and the input differ only in one variable, for which Completeness asserts that the difference in the two output values is equal to the attribution to this variable. Attributions generated by integrated gradients satisfy Implementation Invariance since they are based only on the gradients of the function represented by the network.*

Recall sensitivity (a)

- sensitivity (a)
  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.

Prior literature has relied on empirically evaluating the attribution technique. For instance, in the context of an object recognition task, (Samek et al., 2015) suggests that we select the top k pixels by attribution and randomly vary their intensities and then measure the drop in score. If the attribution method is good, then the drop in score should be large. However, the images resulting from pixel perturbation could be unnatural, and it could be that the scores drop simply because the network has never seen anything like it in training. (This is less of a concern with linear or logistic models where the simplicity of the model ensures that ablating a feature does not cause strange interactions.)

Uniqueness of Integrated Gradients

Not quite understand (for example, why "less of a concern with linear or logistic models")

- A different evaluation technique considers images with human-drawn bounding boxes around objects, and computes the percentage of pixel attribution inside the box. While for most objects, one would expect the pixels located on the object to be most important for the prediction, in some cases the context in which the object occurs may also contribute to the prediction. The cabbage butterfly image from Figure 2 is a good example of this where the pixels on the leaf are also surfaced by the integrated gradients.

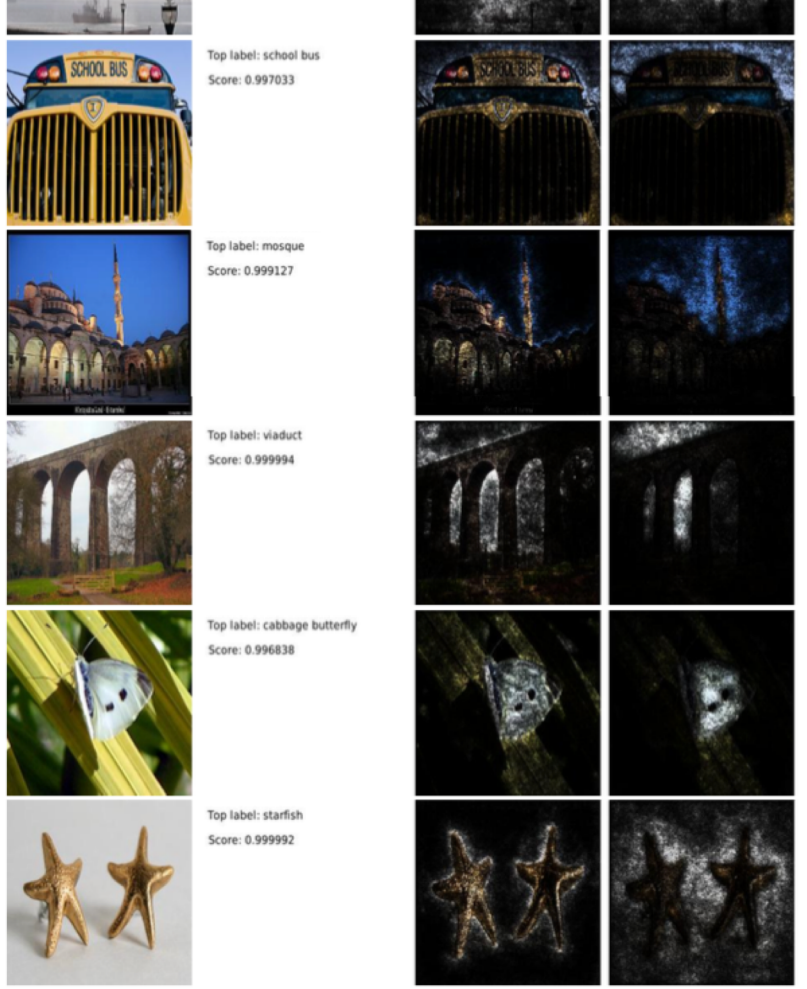

Figure 2. **Comparing integrated gradients with grad the image.** Left-to-right: original input image, label and score for the highest scoring class, visualization of integr dients, visualization of gradients*image. Notice that th izations obtained from integrated gradients are better at r distinctive features of the image.

- Roughly, we found that every? empirical evaluation technique we could think of could not differentiate between artifacts that stem from
  - perturbing the data, a misbehaving model,? and
  - a misbehaving attribution method (maybe the method does not satisfy Axiom 1: sensitivity (a)).

- This was why we turned to an axiomatic approach in designing a good attribution method (Section 2). While our method satisfies Sensitivity and Implementation Invariance (same explanation as chain rule for derivatives), it certainly isn't the unique method to do so.

Uniqueness of Integrated gradients amongst Path methods

- See the paper.

- All path methods satisfy
  - implementation invariance.
  - Completeness => axiom 1 (a).
- More interestingly, path methods are the only methods that satisfy Axiom 1(a), 2 (implementation invariance), 3 (completeness), Axiom 1(b), Axiom 4 (linearity).

Uniqueness of Integrated gradients amongst Path methods

- Theorem 1: Integrated gradients is symmetry preserving

- From the proof, they are swapping x_i,x_j, however, I am unable to go through one part they talk about integrand.

Symmetry-Preserving. Two input variables are symmetric w.r.t. a function if swapping them does not change the function. For instance, x and y are symmetric w.r.t. F if and only if $F(x, y) = F(y, x)$ for all values of x and y. An attribution method is symmetry preserving, if for all inputs that have identical values for symmetric variables and baselines that have identical values for symmetric variables, the symmetric variables receive identical attributions.

E.g., consider the logistic model $\text{Sigmoid}(x_1 + x_2 + \dots)$. x_1 and x_2 are symmetric variables for this model. For an input where $x_1 = x_2 = 1$ (say) and baseline where $x_1 = x_2 = 0$ (say), a symmetry preserving method must offer identical attributions to x_1 and x_2.

Theorem 1: Integrated gradients is symmetry preserving

- Recall the definition of ”path methods”

A. Proof of Theorem 1

Proof. Consider a non-straightline path $\gamma : [0, 1] \to \mathrm{R}^n$ from baseline to input. W.l.o.g., there exists $t_0 \in [0, 1]$ such that for two dimensions i, j, $\gamma_i(t_0) > \gamma_j(t_0)$. Let (t_1, t_2) be the maximum real open interval containing t_0 such that $\gamma_i(t) > \gamma_j(t)$ for all t in (t_1, t_2), and let $a = \gamma_i(t_1) = \gamma_j(t_1)$, and $b = \gamma_i(t_2) = \gamma_j(t_2)$. Define function $f : x \in [0, 1]^n \to R$ as 0 if $\min(x_i, x_j) \leq a$, as $(b - a)^2$ if $\max(x_i, x_j) \geq b$, and as $(x_i - a)(x_j - a)$ otherwise. Next we compute the attributions of f at $x = \langle 1, \ldots, 1 \rangle_n$ with baseline $x' = \langle 0, \ldots, 0 \rangle_n$. Note that x_i and x_j are symmetric, and should get identical attributions. For $t \notin [t_1, t_2]$, the function is a constant, and the attribution of f is zero to all variables, while for $t \in (t_1, t_2)$, the integrand of attribution of f is $\gamma_j(t) - a$ to x_i, and $\gamma_i(t) - a$ to x_j, where the latter is always strictly larger by our choice of the interval. Integrating, it follows that x_j gets a larger attribution than x_i, contradiction. □

Integrated gradients averaging over multiple paths?

- See remark 5 (Shapley-Shubik).
  computationally expensive in deep networks

Applying integrated gradients

- **Selecting a benchmark:**
- A key step in applying integrated gradients is to select a good baseline. *We recommend that developers check that the baseline has a near-zero score*— as discussed in Section 3, this allows us to interpret the attributions as a function of the input. But there is more to a good baseline: For instance, for an object recognition net- work it is possible to create an adversarial example that has a zero score for a given input label (say elephant), by applying a tiny, carefully-designed perturbation to an image with a very different label (say microscope) (cf. (Goodfellow et al., 2015)). The attributions can then include undesirable artifacts of this adversarially constructed baseline. So we would additionally like the baseline to convey a complete absence of signal, so that the features that are apparent from the attributions are properties only of the input, and not of the baseline.

Applying integrated gradients

- For instance, in an object recognition net- work, a black image signifies the absence of objects. The black image isn’t unique in this sense—an image consisting of noise has the same property. However, using black as a baseline may result in cleaner visualizations of “edge” fea- tures. For text based networks, we have found that the all- zero input embedding vector is a good baseline. The action of training causes unimportant words tend to have small norms, and so, in the limit, unimportance corresponds to the all-zero baseline. Notice that the black image corre- sponds to a valid input to an object recognition network, and is also intuitively what we humans would consider ab- sence of signal. In contrast, the all-zero input vector for a text network does not correspond to a valid input; it never- theless works for the mathematical reason described above.

- Computing Integrated gradients: see paper
- Applications:
  - Object recognition network (ORN): good they have improvement in their result
  - Diabetic Retinopathy Prediction
  - Question classification: (exactly what in my mind weeks ago) – attribution is more clear than ORN
  - Neural machine translation: attribution is more clear than ORN (not the probability matrix)
    however, it maps to [0,1]^n; sum of each = prediction score for each word
  - Chemistry models:column