Axiomatic Attribution for Deep Networks

Mukund Sundararajan (Google)
Joint work with Ankur Taly, Qiqi Yan

Why did the network label this image as “fireboat”?

Problem Statement

Attribute a <u>complex</u> deep network's prediction to its input features

E.g. Attribute an Object Recognition Network's prediction to its pixels

E.g. Attribute a diagnostic network's prediction to patient's symptoms, measurements, history

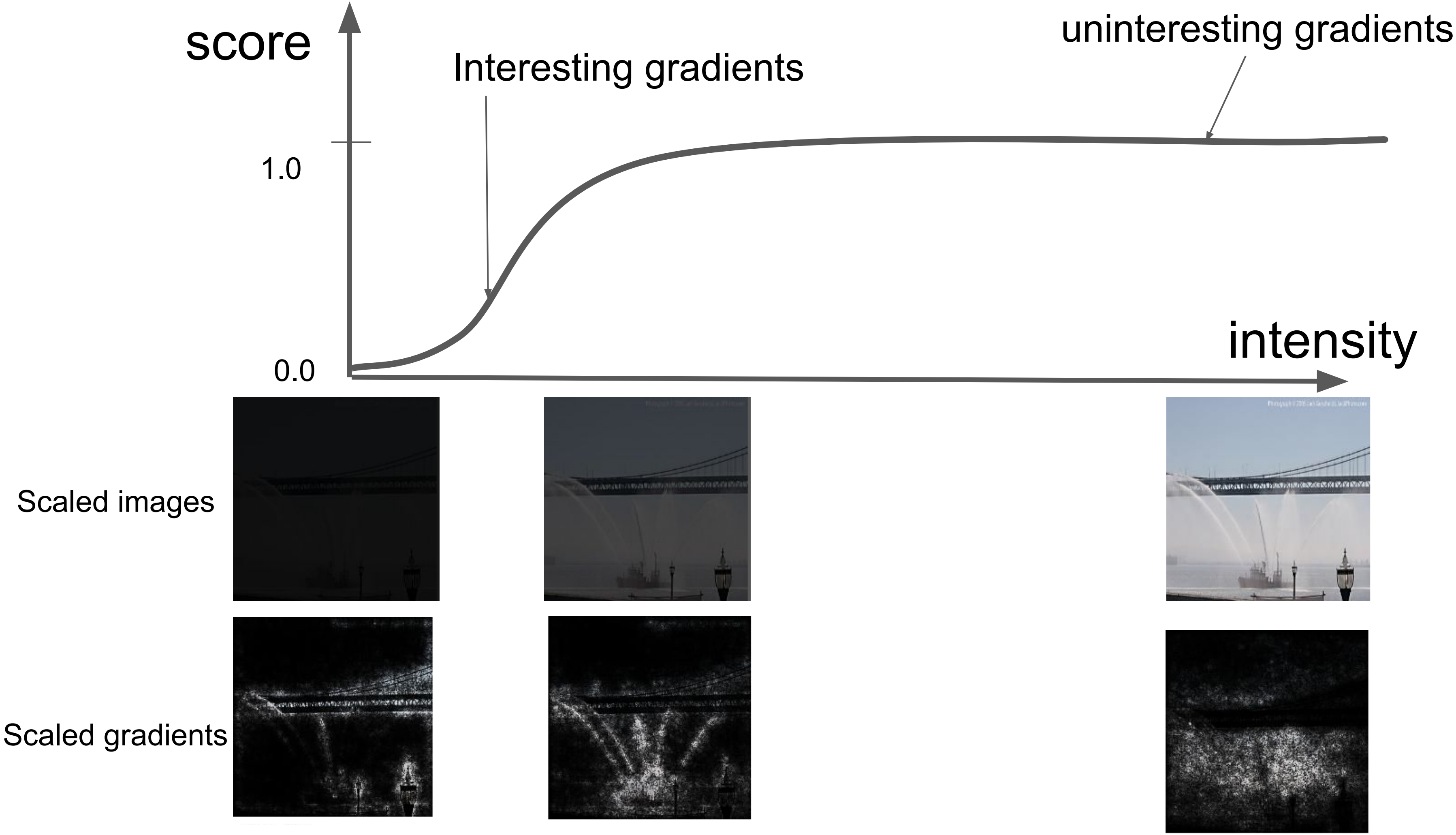
[**DeepLift** '17], [**Layerwise relevance propagation** '16], [**DeConv nets** '14], [**Guided backprop** '14]

Prior work

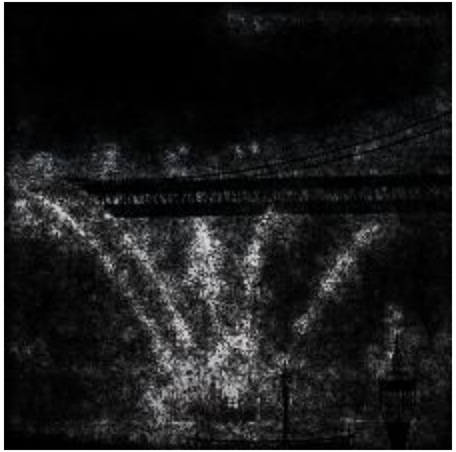
- Prior work is characterized by intuitive design+empirical eval
- Challenges with evaluation
  - Evals from prior work are specific to object recognition
  - **Hard to separate model behavior, attribution errors, eval artifacts**
    - Attributions may “look” incorrect because of unintuitive network behavior
    - Perturbations could change score for artifactual reasons like if they create a “new object” or a “crazy” input

Our approach: Axioms

- **What makes for a good attribution method?**
  - Inspired by economics literature [Aumann-Shapley 1974]
- Overview
  - List **desirable criteria (axioms)** for an attribution method
  - Establish a uniqueness result: X is the **only** method that satisfies these desirable criteria
    - X = Integrated Gradients (our method)
  - E.g. For $f = x*y + z$ and $x,y,z=1,1,1$, the “best” attribution is $attr(x)=0.5$, $attr(y)=0.5$, $attr(z) = 1.0$
- Surprisingly, the resulting method is simple


score
uninteresting gradients
Interesting gradients
1.0
0.0
intensity
Scaled images
Scaled gradients

$$IG(input,base) = (input - baseline)* \int_{0-1} \nabla F(\alpha * input + (1-\alpha) * baseline)\, d\alpha$$



Easy to implement

```
def integrated_gradients(inp, baseline, label, steps=range(50)):
    t_input = input_tensor()  # input tensor
    t_prediction = prediction_tensor(label) # output tensor
    t_gradients = tf.gradients(t_prediction, t_input)[0] # gradients
    path_inputs = [baseline + (i/steps)*(inp-baseline) for i in steps]
    grads = run_network(t_gradients, path_inputs)
    return (inp-baseline)*np.average(grads, axis=0)  # integration
```

Attributions overlaid on the image

Baselines

- **A baseline is an uninformative input** used for comparison
  - E.g. Object recognition: black image, noise image
  - E.g. For f = x*y + z, and attribution at x,y,z=1,1,1, natural baseline is 0,0,0

- Need for a baseline noted by [Deeplift '16], [Layerwise relevance propagation '16]
- Baselines are necessary for attributions [Kahneman-Miller 86]

Axiom: Implementation Invariance

Two networks that compute identical functions <u>for all inputs</u> get identical attributions even if their architecture/parameters differ

E.g. f1 = x*y + z and f2 = y*x + z should get the same attributions

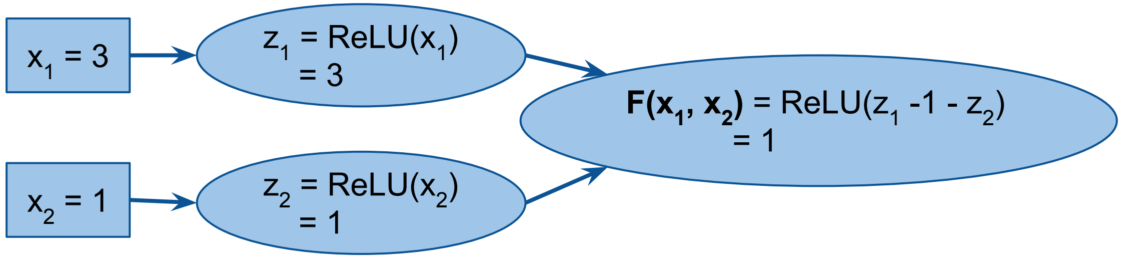
Not satisfied by

- [Deeplift ‘16], [Layerwise relevance propagation ‘16]

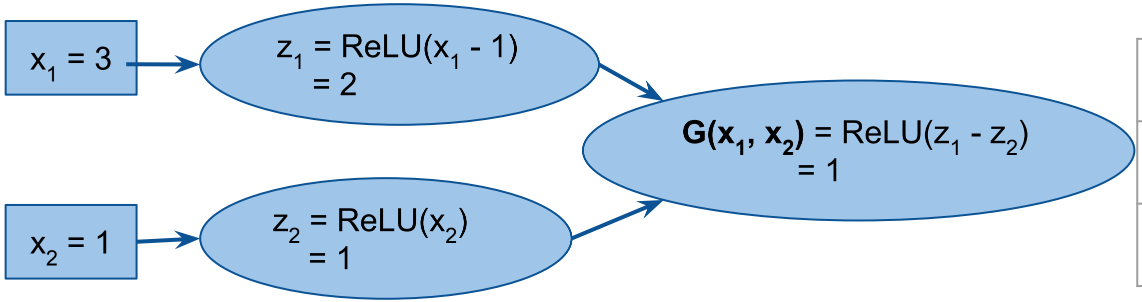
Failure of Implementation Invariance

- [**DeepLift** ‘17], [**Layerwise relevance propagation** ‘16] fail implementation invariance
- Methods redistribute prediction from output to the input over network structure
  - Chain rule is invalid for discrete gradients

For all x_1 and x_2: $F(x_1, x_2) == G(x_1, x_2)$



Integrated gradients	**$x_1 = 1.5$, $x_2 = -0.5$**
DeepLift	$x_1 = 1.5$, $x_2 = -0.5$
LRP	$x_1 = 1.5$, $x_2 = -0.5$



Integrated gradients	**$x_1 = 1.5$, $x_2 = -0.5$**
DeepLift	$x_1 = 2$, $x_2 = -1$
LRP	$x_1 = 2$, $x_2 = -1$

Axiom: Sensitivity

(a) If baseline and input have different scores, but differ in a single variable, then that variable gets some attribution.

(b) If a variable has no influence on a function, then it gets no attribution.

(a) not satisfied by:

- Gradient at output
- [Deconv nets ‘14]
- [Guided back propagation ‘14]

Axiom: Linearity preservation

Attributions(α*f1 + ß*f2) = α*Attributions(f1) + ß*Attributions(f2)

Rationale: Attributions have additive semantics. It is best to respect any existing linear structure.

E.g. For f = x*y + z, the "optimal" attribution should assign blame independently to ‘z’ and ‘x*y’

Axiom: Completeness

sum(attributions) = f(input) - f(baseline)

Theorem [Friedman 2004]

Every method that satisfies Linearity preservation, Sensitivity and Implementation invariance, and Completeness is a path integral of a gradient.

Axiom: Symmetry Preservation

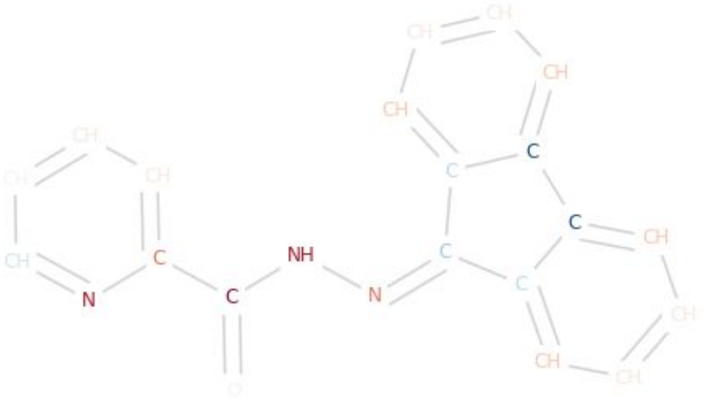
Symmetric variables with identical values get equal attributions

E.g. For $f = x*y + z$, the "optimal" attribution at $x,y,z=1,1,2$ should be equal for x and y.

Theorem: [this work]

Integrated Gradients is the unique path method that satisfies these axioms. (there are other methods that take an average over a symmetric set of paths)

Virtual Drug Screening

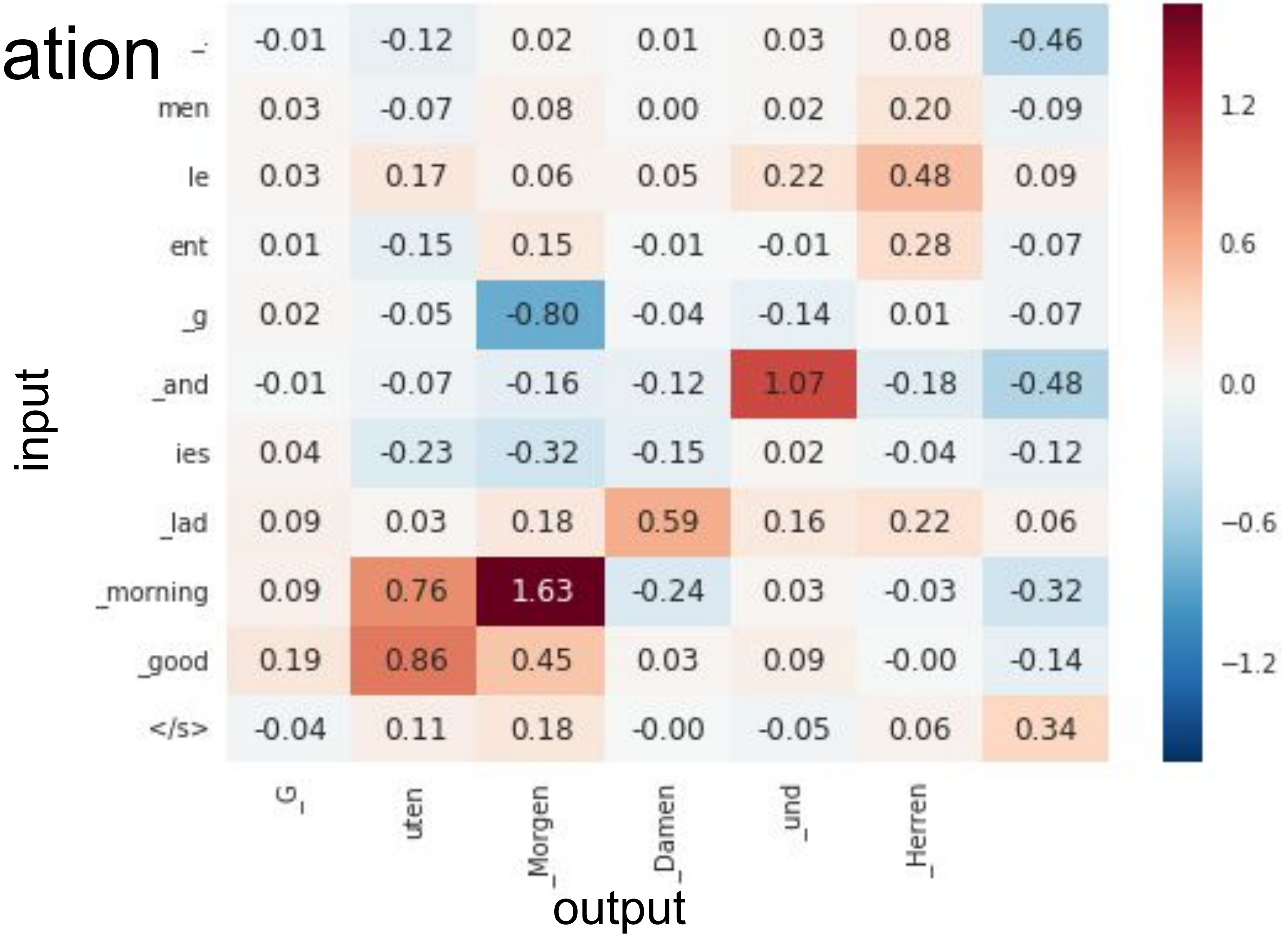


RED: +ve attribution
BLUE: -ve attribtion

Baseline is zero embedding vector

Machine Translation

Attention **does not account** for all the influence of input on output



Reductiveness of “Feature Attribution”

- Suppose that the network predicts TRUE if there are **an even number** of black pixels and FALSE otherwise.

- The attributions to the base features will not tell us much.

Why a vacuum?

Thanks!

We solve the problem of attributing a network's prediction to its base features

Our technique is called Integrated Gradients, is backed by an axiomatic theory, is simple to implement, and applies to a variety of deep networks

https://github.com/ankurtaly/Integrated-Gradients