Visualizing Gradient Attribution Methods for Deep Neural Networks

Pascal Sturmfels
psturm@cs.washington.edu
University of Washington
Seattle, Washington

ABSTRACT

Recent work on understanding the predictions of deep neural networks has converged on local feature attribution methods. Given a prediction, such methods assign credit to each input feature corresponding to how much knowing the value of that feature influenced the prediction. Unfortunately, many existing methods suffer from the same problem based on their mathematical definition: they have a misleading notion of which features are "unimportant". Many of these methods are used as off-the-shelf methods - users may not be aware of the mathematical details concerning why such methods can be misleading. In order to better explain how such methods work, we introduce a set of visualizations that explain the flaws with an existing feature attribution method, integrated gradients. Our visualizations allow users without a mathematical background to understand an inherently mathematical problem with popular feature attribution methods.

1 INTRODUCTION

The rise of neural networks has been driven in large part by their ability to achieve high predictive accuracy on a wide variety of tasks, including state of the art performance on image classification [12, 18], time series prediction tasks [3], and language embedding [4]. Neural networks are often viewed as black boxes: unlike simpler model classes, understanding how neural networks make the predictions they do is a challenging task. One approach is understanding predictions at the local level: given a prediction, how important was each feature in making that prediction?

Such methods are known as feature attribution methods. One of the most popular feature attribution methods for neural networks, called integrated gradients (IG), was proposed by Sundararajan et al. [17]. Given a prediction x, a reference input x' representing feature "missingness", and a network function f, the feature importance of feature i is defined as:

$$\text{IG}_i(x, x') := (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\delta f(x' + \alpha \times (x - x'))}{\delta x_i} \delta\alpha \quad (1)$$

In essence, IG accumulates all of the gradients between features being missing (x') and taking the value at the current input (x) in order to get at how important each feature is. This method is attractive because it is simple to compute and comes with theoretical guarantees [6]. However, buried in the its definition is a problem: how do you define x'? The vector of all 0s is often used to represent missing input, but for a neural network 0 may be a meaningful input. In fact, if 0 is used as a reference, features with the value 0 will never show up as important, even if they are critical from the model's perspective.

This discussion seems like an abstract technicality, and it is hard to convey using words and equations alone. However, it is critical to understanding both integrated gradients and all other feature attributions that depend on some reference x'. In order to better convey the issue, and explain why we propose a different method for feature attributions, we turn towards visualization. In this paper, we take a new approach towards visualizing the feature attributions of the Inception V4 network, a deep convolutional neural network for image classification. In the process, we demonstrate why users should be cautious before using integrated gradients.

2 RELATED WORK

There are many existing methods for doing feature attributions, including layer-wise relevance propagation [1], integrated gradients [17], and SHAP [9]. These methods provide a numerical way to explain neural networks, but are only tangentially related to visualization. There exist several image-specific explanation methods [5, 14, 16] that focus on explaining the output of image models, but do not come with theoretical correctness guarantees. There is also a large body of literature discussing use-cases for feature attribution methods [7, 10, 13].

None of the above are concerned with visualization techniques. There exist many techniques to visualize the internal workings of neural networks [8, 11, 13, 19]. Such techniques take a global approach: they aim to visualize how a network operates over an entire dataset. This work, in contrast, does not aim to develop new ways to visualize neural networks. Instead, it aims to help users understand a particular method for explaining neural networks.

3 METHODS

We visualize which pixels are important to making predictions on the Inception V4 network [18], pre-trained from the

Tensorflow Slim Library [15]. We use the Data-Driven Documents library for all visualizations [2]. In order to visualize which pixels are most important, we plot attribution maps using the white-black color scheme common to [16, 17]. Although this color scheme has less perceptual distinction than others, it allows the user to focus on only the most important pixels in the image, and best highlights the problems with integrated gradients.

Throughout the figures, we visualize equations using images and a dynamic slider. The slider represents the most important free parameter in each equation visualized (typically α from equation (1)), and is linked to the image displays. Our visualizations also feature displays that highlight the parameter x' by allowing the user to select different values of x' via mouse hover. Both interaction techniques unify several different image displays into a coherent visual, and allow the user to understand how the parameters in equation (1) impact the resulting output.

4 RESULTS

Our first interactive figure (Figure 1 - third in the website) demonstrates how to visualize equation (1) without needing background mathematical knowledge. The left-most image visualizes

$$x' - \alpha \times (x - x')$$

where x is an image of a yellow bird, and x' is the image of all zeros, a constant black image. This formula represents interpolating between the bird and a black image at α. Going from left to right, the next image displays

$$(x_i - x_i') \times \frac{\delta f(x' + \alpha \times (x - x'))}{\delta x_i}$$

which shows a single point of the integral, the gradients at that interpolated image. The third figure then displays the cumulative integral up to the chosen value of α:

$$(x_i - x_i') \times \int_{k=0}^{\alpha} \frac{\delta f(x' + k \times (x - x'))}{\delta x_i} \delta k$$

This image simply represents an average of the previous figure over values of $k \leq \alpha$. The slider in the visualization controls α in the equation, and lets the user scrub through each piece of the equation at different points along the integral. This interaction technique allows the user to see how the integral averages over gradients. The chart in the right-most part of the image represents the sum of all pixels in the right-most image. Because we are using an approximation to calculate the integral, the red line represents how good the approximation is - the closer to the output of the network at the original image (in blue), the better the approximation.

Our second interactive figure (the fourth in the website) uses interaction to demonstrate the pitfalls of integrated gradients. Normally, x' is chosen as the all-black image - that means that black represents "missing" pixel information. However, black is a meaningful color in many images, such as in our goldfinch. Depending on which color we use to model missingness, which pixels integrated gradients highlights as important changes. In (Figure 2), we display the interactive visualization. The user can hover over the segmented image to highlight a particular color. All four images are dynamically linked to the user's selection: as the user selects different colors in the segmented image, the figures update to show which pixels are important to the neural network relative to that color. The interactivity in this figure highlights a key challenge with integrated gradients: which color should we use as a reference, and why? How users interpret their models depends on this question, but it has no clear answer.

The third figure we feature in this paper is the fifth in the website, located here as Figure 3. It uses similar techniques to the first figure. Instead of using black as a reference, we propose to use an expectation over the dataset. This method is called expected gradients - a more thourough treatment of it can be found in the SHAP package. Our final figure demonstrates how we avoid specifying a choice of reference, and instead use images from the training dataset to specify "missingness": it is akin to simply drawing randomly from the dataset (an expectation).

5 DISCUSSION

In this work, we introduced a new way to visualize the equation behind a popular feature attribution method. The visualization technique including breaking the equation into chunks related to how the equation is computed, and dynamically linking free parameters of the equation to interactive elements in the display, such as sliders and mouse events. Critically, our visualizations demonstrated a flaw with an existing feature attribution method: the choice of reference is arbitrary, but can drastically change the interpretation of the network. Finally, we used the same visualization techniques to demonstrate how we can fix this flaw.

6 FUTURE WORK

There are many ways to improve the interactive visualizations displayed on our website. Currently, the figures are presented all on one page with accompanying text. However, the figures would be better displayed using some sort of interactive click-through tutorial that presents information as needed, and directs the user's focus to different display elements one at a time. Although the gray-scale color scheme is the de-facto color scheme for plotting pixel-wise attribution maps, it intentionally obscures smaller magnitude effects. Experimentation with the colors used to plot

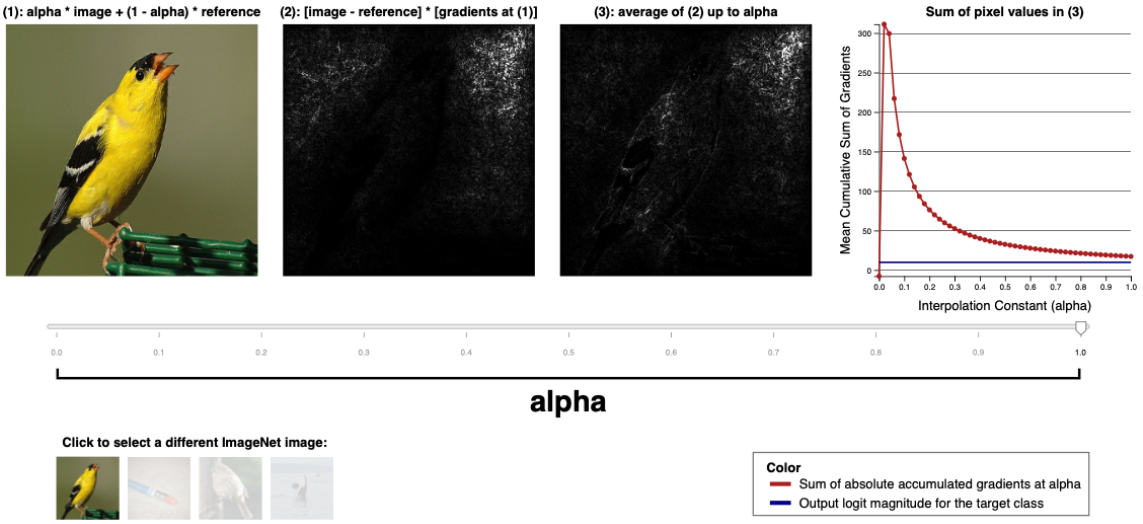

Figure 1: A screenshot of an interactive visualization showing how integrated gradients determines which pixels are most important in making a prediction.

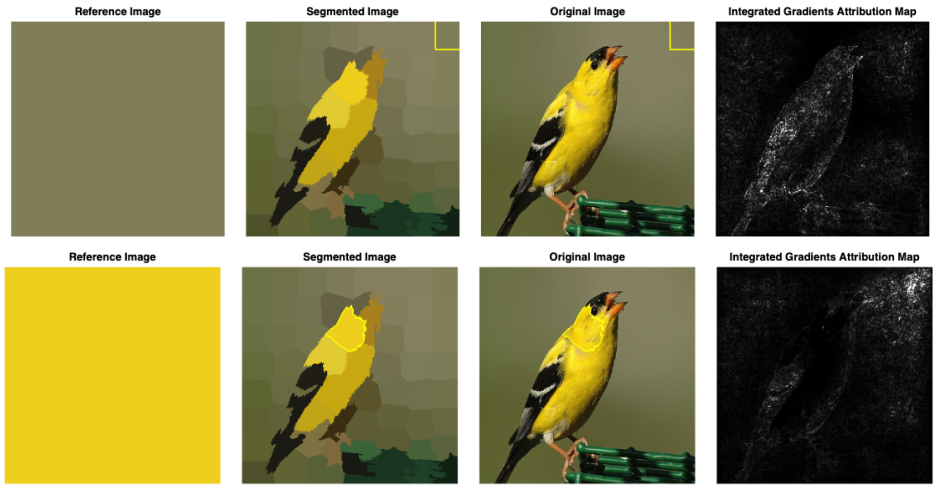

Figure 2: The choice of reference in integrated gradients matters.

such maps may reveal better color schemes for interpretation. Finally, the website may benefit from tactfully-placed, properly-formatted equations.

7 ACKNOWLEDGMENTS

The visualizations in this work describe recent work we (my lab colleagues and I) recently submitted to a conference. That work was in large part aided by Scott Lundberg, who also deserves thanks for ideas and advice regarding this visualization. Thanks as well to Joe Janizek and Gabe Erion, who were responsible for as much of the conference paper as I was, including experimental results and the write-up. Similarly, thanks to my advisor Su In Lee for feedback regarding the paper.

REFERENCES

[1] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one* 10, 7 (2015), e0130140.

[2] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D^3 data-driven documents. *IEEE transactions on visualization and computer graphics* 17, 12 (2011), 2301–2309.

[3] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. 2018. Recurrent neural networks for multivariate time series with missing values. *Scientific reports* 8, 1 (2018), 6085.

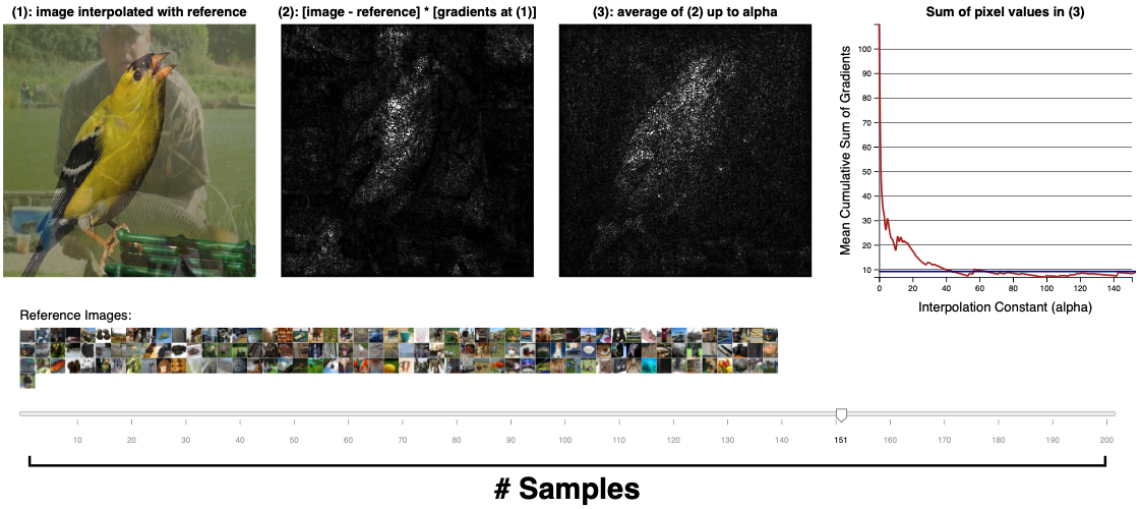

Figure 3: A screenshot of our proposed method: expected gradients.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[5] Ruth C Fong and Andrea Vedaldi. 2017. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*. 3429–3437.

[6] Eric J Friedman. 2004. Paths and consistency in additive cost sharing. *International Journal of Game Theory* 32, 4 (2004), 501–518.

[7] Hyunkwang Lee, Sehyo Yune, Mohammad Mansouri, Myeongchan Kim, Shahein H Tajmir, Claude E Guerrier, Sarah A Ebert, Stuart R Pomerantz, Javier M Romero, Shahmir Kamalian, et al. 2019. An explainable deep-learning algorithm for the detection of acute intracranial haemorrhage from small datasets. *Nature Biomedical Engineering* 3, 3 (2019), 173.

[8] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2018. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*. 6389–6399.

[9] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*. 4765–4774.

[10] Scott M Lundberg, Bala Nair, Monica S Vavilala, Mayumi Horibe, Michael J Eisses, Trevor Adams, David E Liston, Daniel King-Wai Low, Shu-Fang Newman, Jerry Kim, et al. 2018. Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. *Nature Biomedical Engineering* 2, 10 (2018), 749.

[11] Aravindh Mahendran and Andrea Vedaldi. 2016. Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision* 120, 3 (2016), 233–255.

[12] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision* 115, 3 (2015), 211–252.

[13] Rory Sayres, Ankur Taly, Ehsan Rahimy, Katy Blumer, David Coz, Naama Hammel, Jonathan Krause, Arunachalam Narayanaswamy, Zahra Rastegar, Derek Wu, et al. 2019. Using a deep learning algorithm and integrated gradients explanation to assist grading for diabetic retinopathy. *Ophthalmology* 126, 4 (2019), 552–564.

[14] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*. 618–626.

[15] Nathan Silberman and Sergio Guadarrama. 2016. Tensorflow-slim image classification model library. (2016).

[16] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. 2017. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825* (2017).

[17] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 3319–3328.

[18] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[19] Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*. Springer, 818–833.