Revisiting Sanity Checks for Saliency Maps

Anonymous Author(s)
Affiliation
Address
email

Abstract

Saliency methods are a popular approach for model debugging and explainability. However, in the absence of ground-truth data for what the correct maps should be, evaluating and comparing different approaches remains a long-standing challenge. The sanity checks methodology of Adebayo et al [Neurips 2018] has sought to address this challenge. They argue that some popular saliency methods should not be used for explainability purposes since the maps they produce are not sensitive to the underlying model that is to be explained. In this work we revisit the logic behind their proposed methodology. We cast the objective of ruling out a saliency method as not being sensitive to the model as a causal inference question, and use this to argue that their empirical results do not sufficiently establish their conclusions, due to a form of confounding that may be inherent to the tasks they evaluate on. On a technical level, our findings call into question the current consensus around some methods being not suitable for explainability purposes. On a broader level, our work further highlights the challenges involved with saliency map evaluation.

1 Introduction

Feature attribution methods are designed to highlight the features in the input that are important for a given model's predictions. This can be useful for both interpretability and debugging, as a human expert can probe these attributions and verify the extent to which they align with their own priors. In this work, we focus on feature attributions methods in the domain of computer vision, where they are broadly referred to as *saliency maps*. Perhaps the most straightforward approach is to compute the saliency map by deriving the gradient of the model's predicted class w.r.t the input image [5, 3]. We refer to this approach as *vanilla backpropogation*. In recent years, a wealth of different variations of this basic method have been proposed [8, 6]; for example, *guided backpropagation* (GBP) [7] modifies the gradient computation such that the gradients of negative RELUs are zeroed out. While saliency maps produced using GBP look visually more appealing than those produced with VBP, there is no clear sense in which the former are indeed "better" the latter. Indeed, evaluating performance of the different proposed methods (as well as measuring progress) remains a fundamental challenge since in general, the task of saliency maps inherently lacks "ground truth".

In recent work, Adebayo *et al.* [1] addressed this challenge by proposing several basic "sanity checks" for saliency methods. The simple idea behind the derivation of their tests is that in so far model debugging and explainability is the objective, using saliency methods that are not sensitive to the underlying model is a bad choice. A classic example are edge detection techniques: while they may produce visually-appealing saliency maps, they are useless for model debugging as the edges depend only on the specific image in question. To test whether a particular saliency method S is sensitive to the underlying model, they compare the saliency map on an image x with a saliency map on the same image, with the exception that the underlying model is replaced: e.g., instead of the model M, a perturbed model is used (say, M', a randomly initialized network of the same architecture). If the method depends on the model in question, one would expect the two maps $S(x; M)$ and $S(x; M')$

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

to be *different*; hence, if they are observed to be *similar*, then we conclude that the method is not sensitive to the model in question. In their work, [1] apply this methodology to standard pre-trained models on MNIST and ImageNet. Their conclusion is that some saliency methods (including VBP) pass these sanity checks, while others (including GBP) do not.

This work: a critical perspective. We revisit the sanity checks methodology of [1]. Our main point criticism is that they only established $S(x; M) \approx S(x, M')$ when the image x and trained model M were derived from a particular task (e.g., x is a image of a dog and M is a model trained on ImageNet). We argue that this constitutes a form of *selection bias* that could potentially invalidate the conclusion that the method S is itself not sensitive to the model choice M. For example, one could hypothesize that the early layers of the untrained, randomly-initialized model M' act as sufficiently strong object detectors, in that they are able to localize the object in the image (of course, without knowing it is a dog). If the trained model M truly attends to the dog in the image, then the "correct" saliency map would highlight the pixels of the dog – which in principle, both M and M' could do. The fact that in *this particular instance* $S(x; M) \approx S(x, M')$ seems insufficient to rule out the saliency method S as being insensitive to the model choice. At the very least, we would like to establish $S(x; M) \approx S(x, M')$ also for images and models for which we have strong reasons to believe that the above hypothesis could not, on its own, explain the similarity $S(x; M) \approx S(x, M')$.

Slightly more formally, our discussion above highlighted one potential form of confounding that should be eliminated if we were to establish the claim that the model choice truly does not effect the saliency output. Building on the above intuition, we design two custom tasks (one derived from MNIST, the other from ImageNet) that are specifically tailored to eliminate this source of confounding. We demonstrate that performing the sanity checks w.r.t these tasks leads to very different conclusions than those of Adebayo et al [1]; in particular, *both VBP and GBP pass the tests*. On a technical level, this finding calls into question the current consensus around GBP being less-suitable for model debugging purposes [2, 1]. On a broader level, our work further highlights the challenges in saliency method evaluation, as it reveals that even seemingly simple sanity checks can be misleading if their derivation does not fully account for important background conditions. We hope that our observations, albeit simple, will inspire further research into these questions that will enable a more reliable evaluation of different saliency methods and a better understanding of their differences.

The rest of this manuscript is organized as follows. We begin with some preliminaries in Section 2. In Section 3 we formally define the custom tasks mentioned above and demonstrate that in this case, $S(x; M) \neq S(x, M')$ for both VBP and GBP. In Section 4 we discuss an interpretation of our proposed methodology from a causal inference perspective, that tries to formalize the idea of eliminating confounding factors when establishing the effect of model choice on saliency output.

2 Preliminaries

Saliency methods. A model is mapping from an image $x \in \mathbb{R}^d$ to a prediction over k-classes, which we interpret as a vector $M(x) \in \mathbb{R}^k$. A saliency method is a mapping from images $x \in \mathbb{R}^d$ to *saliency maps*, which are also images in $\mathbb{R}^d$. Since the method may depend on the model in question, we use $S(x; M) \in \mathbb{R}^d$ to denote the resulting map. Since our focus is on computer vision, and the models in question are typically deep learning models (and hence differentiable), our focus is on gradient-based saliency methods, namely methods in which we use backpropogation to compute the saliency map. Specifically, we restrict our attention to two commonly used methods: vanilla backpropogation and guided backpropogation. The former simply computes the saliency map as the derivative of the predicted class w.r.t the input image: $S^{\mathsf{VBP}}(x; M) = \frac{\partial \max_j M(x)_j}{\partial x}$; Guided backpropagation $S^{\mathsf{GBP}}(x; M)$ is a modification that sets negative entries to zero when backproping through RELU units [7]. Guided back-propogation tends to produce more visually-appealing saliency maps. Our restriction to these two methods is motivated by the fact that despite their similarity, the sanity checks of [1] clearly distinguished them: GBP failed their tests, while VBP passed them.

Causal graphs. Causal inference allows us to formalize the effect of hypothetical actions (interventions), such as the effect of setting the model $\mathbf{M}$ to be M or M' on the resulting saliency map. In this work we employ the language of causal graphs. A causal graph is a directed a-cyclic graph in which edges are endowed with a causal interpretation; for a full presentation of the ideas we refer the

reader to Pearl's primer [4]. The causal effect of an action $X = x$ on an outcome Y is written using a special *do* operator, $\mathbf{Pr}\left[Y = y|do(X = x)\right]$. A fundamental question in causal inference is when can equations that include the do-operator be re-written in terms of observed variables. In what follows we will be using the *back-door* criterion, which we state below.

Back-door criterion (Definition 3.3.1 in [4]):

- Given an ordered pair of variables (X, Y) in a directed acyclic graph G, a set of variables Z satisfies the backdoor criterion relative to (X, Y) if no node in Z is a descendant of X, and Z blocks every path between X and Y that contains an arrow into X.
- If Z satisfies the back-door criterion w.r.t (X, Y), then the causal effect of X on Y is given by the formula $\mathbf{Pr}\left[Y = y|do(X = x)\right] = \sum_z \mathbf{Pr}\left[Y = y|X = x, Z = z\right] \cdot \mathbf{Pr}(Z = z)$.

As we will see, we will focus on the case in which the treatment variable X is binary and we are interested in the average causal effect, $\mathbf{E}\left[Y|do(X = 1)\right] - \mathbf{E}\left[Y|do(X = 0)\right]$.

3 Experiments

We refer to a *task* as a joint distribution over images and labels. In this section we derive two tasks, one from MNIST and one from ImageNet, which are specifically tailored to ensure that the early-layer priors of an untrained network will be different from those of a trained network. In both cases we achieve this by creating *multi-object* images with labels that depend only on one object. We show that for the images of this task, $S(x; M)$ is very different from $S(x; M')$ for both GBP and VBP.

3.1 MNIST

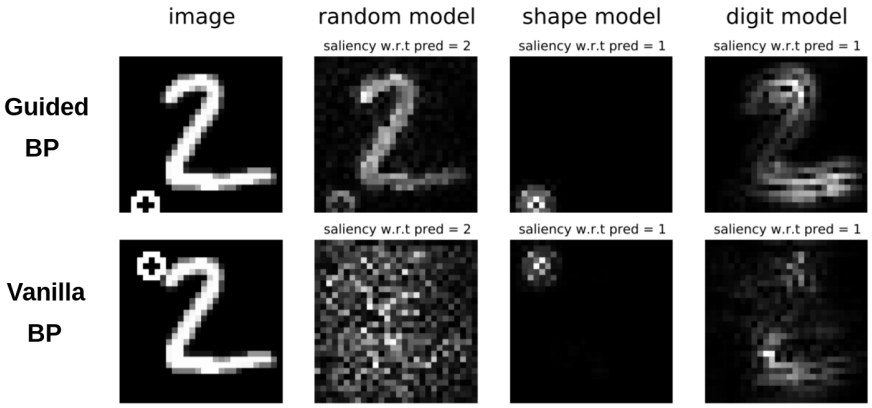
Task description. For every MNIST image, we randomly perform one of the following three manipulations: (i) add a random rectangle, (ii) add a random empty-circle shape, (iii) do nothing. See Figure 1 for an illustration. We then construct two different classification tasks (labeled datasets): `ShapeWithMnist` in which the label is $\{0, 1, 2\}$ (depending on which shape is present in the input) and `MnistWithShape` in which the label is $\{0, 1\}$ depending on whether the digit in the image is smaller than 5 or not. We continue to train two models, a `shape` model on the `ShapeWithMnist` data, and a `digit` model on the `MnistWithShape` data, both to a test accuracy of 0.99.

Figure 1: Random examples from our manipulated MNIST task.

Results. In Figure 2 we demonstrate example saliency maps generated w.r.t each model's predicted label using either GBP (top row) or VBP (bottom row). The *random model* column (second to the left) represents $S(x; M')$ whereas the two right-most columns represent $S(x; M_1)$ and $S(x; M_2)$, respectively. It's interesting to note that when applied to the random model, GBP (unlike VBP) indeed produces saliency maps that resemble edge detection, a kin to the observations of [1]. However, our main point is that this is not a criticism of GBP as a saliency method, since we clearly see that $S(x; M') \neq S(x; M_1), S(x; M_2)$. Furthermore, both models produce correct saliency maps that highlight the relevant object in the image.[1] We refer the reader to Appendix A for additional examples for each method.

[1] We note that we only know the "correct" maps because in this case we engineered the tasks such that high accuracy is only possible when the model learns to attend to one type of object and ignore the other. In general, as mentioned earlier, we have no knowledge of the correct saliency maps and the sanity checks can serve as one proxy for eliminating saliency methods which are clearly problematic.

Figure 2: **Sanity checks performed w.r.t to our manipulated MNIST task**. The three columns on the right show the saliency maps produced for the different models; from right to left: a model trained on the `ShapeWithMnist`, a model trained on the `MnistWithShape` task, and a random model of the same architecture. Both methods provide accurate maps (highlighting the object that determines the label) that are visually distinct from the maps of the random model.



3.2 ImageNet

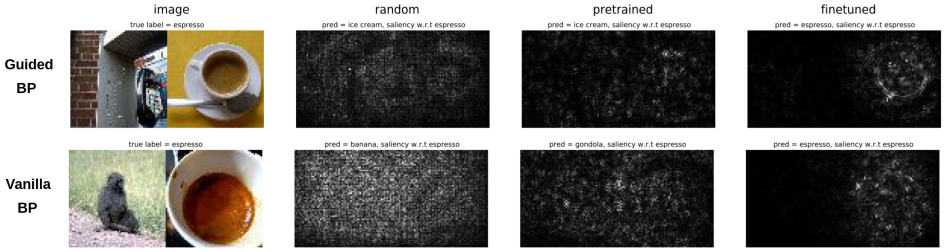
Task description. We generate custom pairs of ImageNet images, as follows. First, we consider the TinyImageNet dataset, which consists of 250 training images and 50 validation images for each original ImageNet class, downscaled to 64x64. We restrict our attention to ten ImageNet classes, and divide them into two groups of five: the first group consists of the labels [acorn, espresso, banana, ice-cream, gondola] and the second consists of the labels [wok, cauliflower, space heater, payphone, baboon]. We then construct a new dataset as follows: we randomly select one image from the first group (e.g. an espresso) and one image from the second group (e.g. a baboon), where the position is selected randomly. The label of this image pair is defined to be the label of the image that belongs to the first group. We finetune a pretrained ResNet18 model on our pairs dataset until the model reaches a test accuracy of around 0.77.

Results. In Figure 3 we demonstrate example saliency maps generated w.r.t the true label using either GBP (top row) or VBP (bottom row). The *random model* column (second to the left) and the pre-trained column (third to the left) represent $S(x; M_1')$ and $S(x; M_2')$, whereas the right-most column represents $S(x; M)$. Again, we visually see that the two baseline models have positive saliency for both regions of the image, as expected, whereas the trained model has positive saliency only on the relevant part of the image (in this case, the espresso, which determines the class label). That is, again we demonstrate that $S(x; M_1'), S(x; M_2') \neq S(x; M)$. We refer the reader to Appendix B for additional examples.

4 Interpretation

In this section we attempt to use the language of causality to re-interpret the sanity check methodology and to further motivate the custom tasks defined in Section 3. Recall that the initial objective of [1] is to determine whether a particular saliency method is sensitive to the underlying model. In causal terms, this is like asking whether the *individual-level causal effect* of the model choice on the output saliency map is zero. We therefore draw a causal graph for this setup by assuming a binary node $\mathbf{M}$ (where 1 denotes using the trained model M and 0 denotes using the baseline model M'), a node $\mathbf{X}$ representing the random variable for the image in question and a third node $\mathbf{S}$ representing the random variable for the resulting saliency map, for a given saliency method S. Since the saliency output naturally depends on both the model and the image we draw arrows $\mathbf{M} \to \mathbf{S}, \mathbf{X} \to \mathbf{S}$. We cannot end there, since $\mathbf{X}$ and $\mathbf{M}$ are currently independent, which we know is not the case; we capture this by introducing a fourth node, $\mathbf{T}$, representing the *task* in question, and including both $\mathbf{T} \to \mathbf{X}$ and $\mathbf{T} \to \mathbf{M}$. This results in a causal graph G which can be thought of as specifying some (unknown) structural causal

Figure 3: **Sanity checks performed w.r.t to our manipulated ImageNet task**. The three columns on the right show the saliency maps produced for the different models. From right to left: a ResNet18 model finetuned on the task in question, a pretrained ResNet18 without finetuning, and a random ResNet18. Both methods provide saliency maps that focus only on the relevant image (in this case, the espresso), whereas the other models have positive saliency for both regions of the image, as expected.



model which governs the behaviour of the saliency method S. Our objective is to determine the causal effect of $\mathbf{M}$ on $\mathbf{S}$ in this model: $\mathbf{E}\left[S|X=x, do(\mathbf{M}=1)\right] - \mathbf{E}\left[S|X=x, do(\mathbf{M}=0)\right]$, where the expectation is w.r.t any randomness in the saliency method. Replacing the do-operation with regular conditional expectation (and assuming S is deterministic, only for simplicity) would amount to evaluating $S(x;M) - S(x;M')$, which is the sanity check of [1]. However, this would be incorrect for the graph we drew, since $\mathbf{M}$ and $\mathbf{S}$ are confounded: there is a back-door path $\mathbf{M} \leftarrow \mathbf{T} \rightarrow \mathbf{X} \rightarrow \mathbf{S}$. Informally, Pearl's do-calculus instructs us to "close" all back-door paths to compute the causal effect of $\mathbf{M}$ on $\mathbf{S}$; in our case this is done by conditioning on the fork, $\mathbf{T}$. Formally, $\{\mathbf{T}\}$ satisfies the back-door criterion w.r.t $(\mathbf{M}, \mathbf{S})$ (see Section 2 for a formal definition) and so the causal effect should be evaluated as follows:

$$\sum_t \left(\mathbf{E}\left[S(x;M)\right] - \mathbf{E}\left[S(x;M')\right]\right) \cdot \mathbf{Pr}(T=t|X=x) \tag{1}$$

where in the we replaced $\mathbf{E}\left[S|X=x, T=t, M=1\right]$ (similarly, $M=0$) with $\mathbf{E}\left[S(x;M)\right]$ since the dependence on T is already captured in $X=x$. Naturally, this expression depends on the specific distribution that the root node $\mathbf{T}$ takes in the structural causal model in question. If, for example, $\mathbf{T}$ is a uniform distribution over the standard MNIST task and the manipulated MNIST task, and x is an image sampled from the manipulated MNIST task, then the conditional distribution $\mathbf{Pr}(T=t|X=x)$ obtains all its mass on the manipulated task and so Equation (1) amounts to the test we performed in Section 3.

A Additional results: manipulated MNIST

Figure 4: Additional examples: saliency maps produced using vanilla backprop.

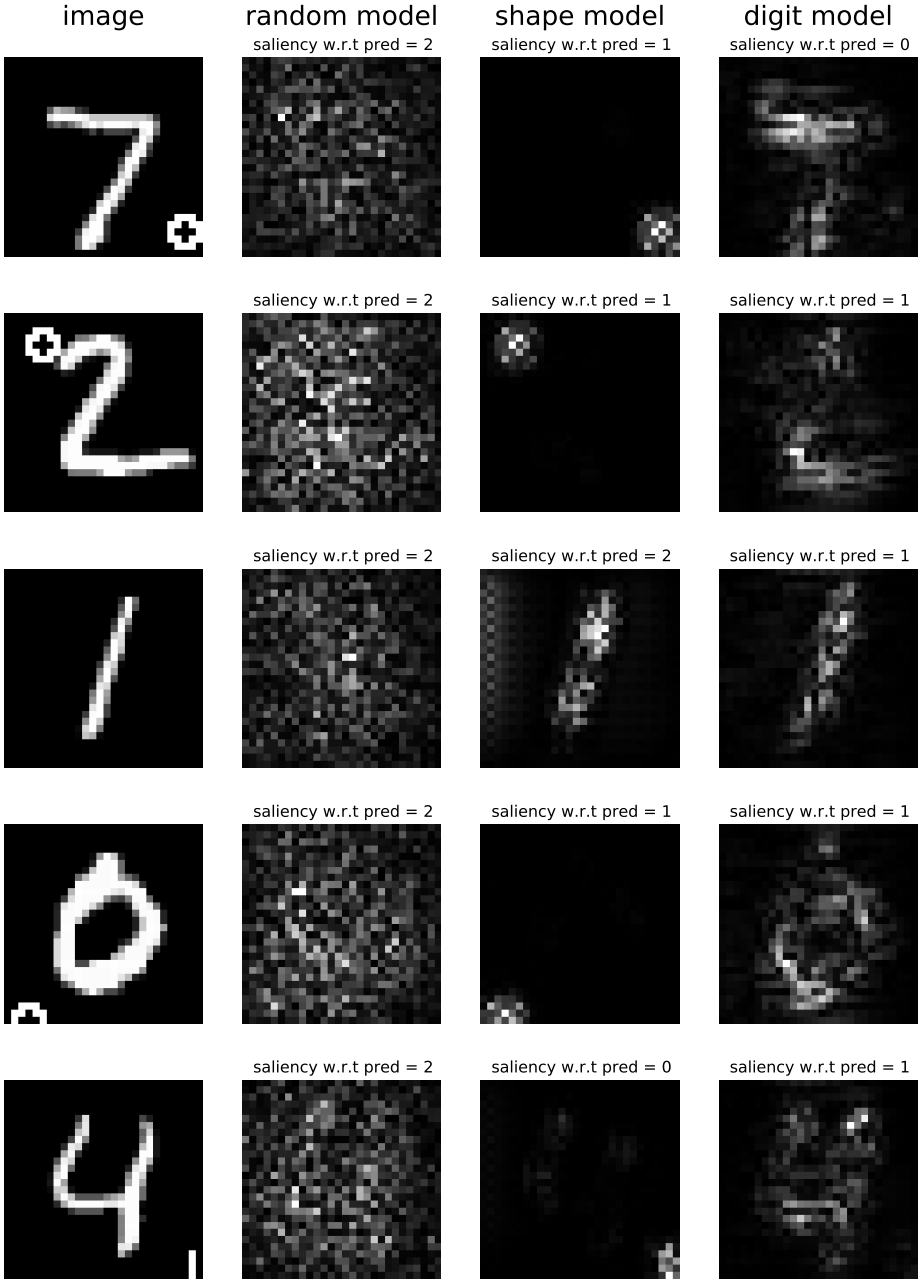
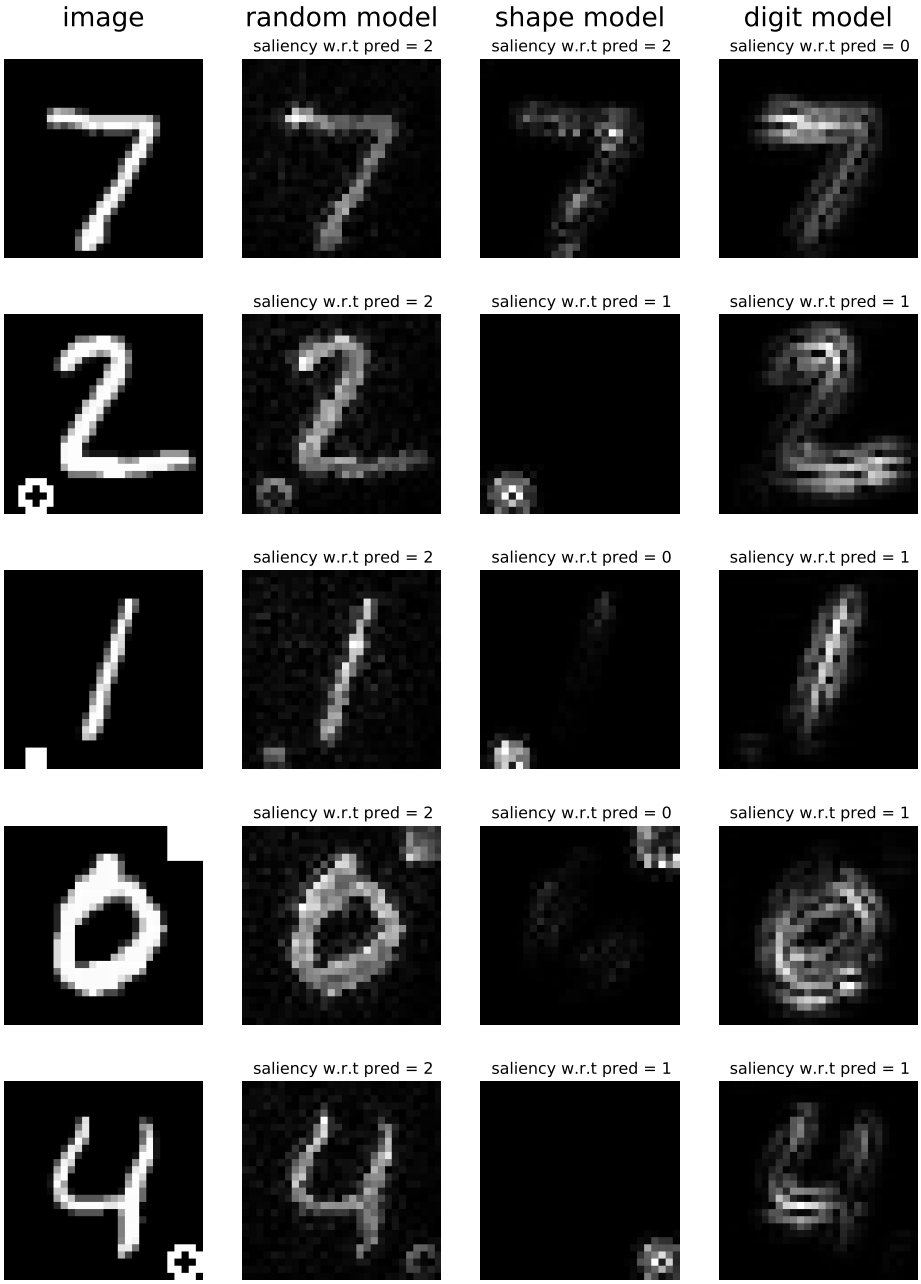


Figure 5: Additional examples: saliency maps produced using guided backprop.



B Additional results: manipulated ImageNet

Figure 6: Additional examples: saliency maps produced using vanilla backprop.

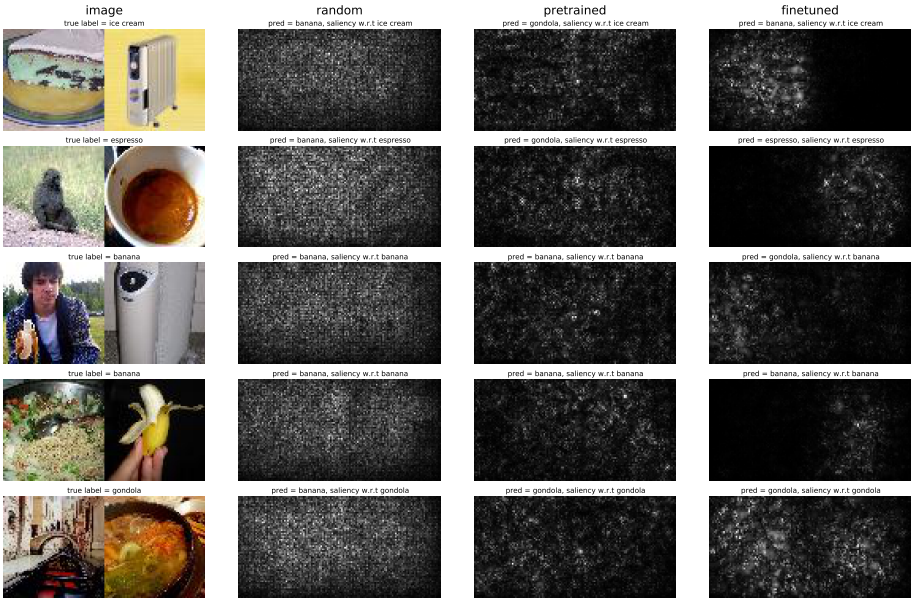
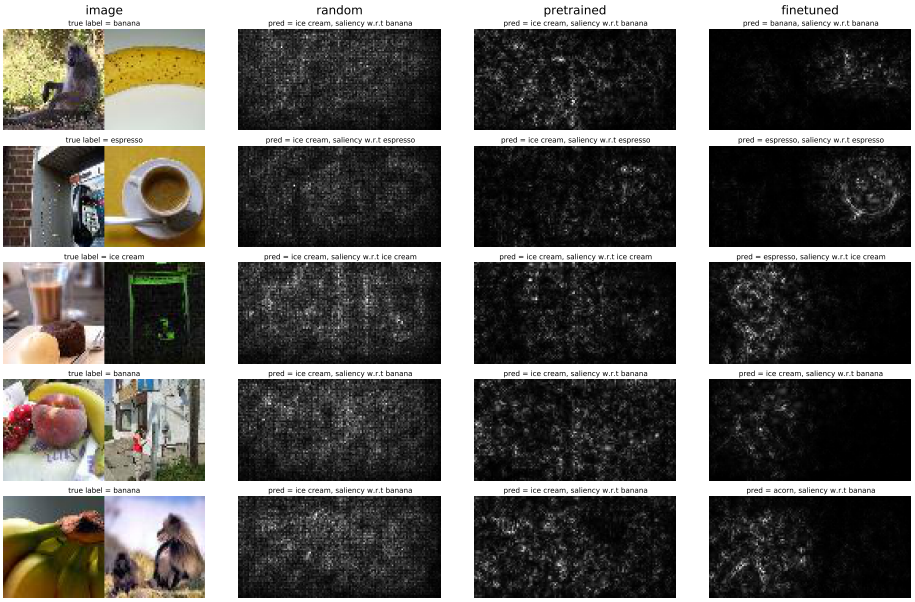


Figure 7: Additional examples: saliency maps produced using guided backprop.



References

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *arXiv preprint arXiv:1810.03292*, 2018.

[2] Julius Adebayo, Michael Muelly, Ilaria Liccardi, and Been Kim. Debugging tests for model explanations. *arXiv preprint arXiv:2011.05429*, 2020.

[3] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

[4] Judea Pearl, Madelyn Glymour, and Nicholas P Jewell. *Causal inference in statistics: A primer*. John Wiley & Sons, 2016.

[5] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

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

[7] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[8] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR, 2017.