A Theoretical Explanation for Perplexing Behaviors of Backpropagation-based Visualizations

Weili Nie [1] **Yang Zhang** [2] **Ankit B. Patel** [1 3]

Abstract

Backpropagation-based visualizations have been proposed to interpret convolutional neural networks (CNNs), however a theory is missing to justify their behaviors: Guided backpropagation (GBP) and deconvolutional network (DeconvNet) generate more human-interpretable but less class-sensitive visualizations than saliency map. Motivated by this, we develop a theoretical explanation revealing that GBP and DeconvNet are essentially doing (partial) image recovery which is unrelated to the network decisions. Specifically, our analysis shows that the backward ReLU introduced by GBP and DeconvNet, and the local connections in CNNs are the two main causes of compelling visualizations. Extensive experiments are provided that support the theoretical analysis.

1. Introduction

Driven by massive data and computational resources, modern convolutional neural networks (CNNs) and other network architectures have achieved many outstanding results, such as image recognition (Krizhevsky et al., 2012), neural machine translation (Sutskever et al., 2014), and playing Go games (Silver et al., 2016), etc. Despite their extensive applications, these neural networks are always considered as black boxes. Interpretability used to be for its own sake; now, due to safety-critical applications such as self-driving cars and tumor diagnosis, it is no longer satisfying to have a black box that is unaccountable for its decisions. The demand for explainable artificial intelligence (XAI) (Gunning, 2017) – human interpretable explanations of model decisions – has driven the development of visualization techniques, including image synthesis via activation maximization (Simonyan et al., 2013; Johnson et al., 2016; Nguyen et al., 2016) and backpropagation-based visualizations (Simonyan et al., 2013; Zeiler & Fergus, 2014; Springenberg et al., 2014; Shrikumar et al., 2017; Kindermans et al., 2017).

The basic idea of backpropagation-based visualizations is to highlight class-relevant pixels by propagating the network output back to the input image space. The intensity changes of these pixels have the most significant impact on network decisions. Specifically, (Simonyan et al., 2013) visualizes the spatial support of a given class in a given image, i.e. saliency map, by using the true gradient which masks out negative entries of bottom data via the forward ReLU. Despite its simplicity, the results of saliency map are normally very noisy which makes the interpretation difficult. (Zeiler & Fergus, 2014) visualize the reverse mapping from feature activities back to the input pixel space with the deconvolutional network (DeconvNet) method. The basic idea of DeconvNet is to mask out negative entries of the top gradients by resorting to the backward ReLU. (Springenberg et al., 2014) proposed the Guided Backpropagation (GBP) method which combines the above two methods: by considering both the forward and backward ReLUs, it masks out the values for which either top gradients or bottom data are negative and produces sharper visualizations. More recently, DeepLift (Shrikumar et al., 2017) and PatternNet (Kindermans et al., 2017) have been proposed to further improve the visual quality of backpropagation-based methods.

This class of backpropagation-based visualizations, in particular GBP and DeconvNet, has attracted a lot of attention in both the deep learning community and other fields (Szegedy et al., 2013; Dosovitskiy & Brox, 2016; Selvaraju et al., 2016; Fong & Vedaldi, 2017; Kraus et al., 2016). Despite their good visual quality, the question of how they are actually related to the decision-making has remained largely unexplored. Do the pretty visualizations actually tell us reliably about what the network is doing internally? Our experiments have confirmed previous observations (Mahendran & Vedaldi, 2016; Selvaraju et al., 2016; Samek et al., 2017) that saliency map is indeed very sensitive to the change of class labels, while GBP and DeconvNet, though their visualization results are much cleaner than saliency map, remain almost the same given different class labels. It seems that

[1]Department of Electrical and Computer Engineering, Rice University, Houston, USA. [2]Department of Computer Science, Rice University, Houston, USA. [3]Department of Neuroscience, Baylor College of Medicine, Houston, USA. Correspondence to: Weili Nie <wn8@rice.edu>, Ankit B. Patel <abp4@rice.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

the visual quality improvement of backpropagation-based methods is sacrificing the ability of highlighting important pixels to a specific output class. In this sense, GBP and DeconvNet may be unreliable in interpreting how deep neural networks make classification decisions.

The most commonly used explanation for these visualizations is to approximate the neural networks with a linear function (Simonyan et al., 2013; Kindermans et al., 2017), where the derivative of output with respect to input image is just the weight vector of the model. In such sense, the backpropagation-based methods can be regarded as visualizing the *learned weights*. But apparently the approximate linear model is too simplistic to reflect the highly nonlinear property of deep neural networks. For example, GBP and DeconvNet essentially apply the same algorithm as saliency map, but treat ReLU, the nonlinear activation, differently. The linear model explanation thus cannot answer questions regarding why GBP and DeconvNet outperform saliency map in terms of visual quality whereas they are less class-sensitive than saliency map, as both of them reduce to saliency map in a linear model. Therefore, we need a more complex model, which should at least capture the impact of both forward ReLU and backward ReLU, to better understand what the main causes of their visually compelling results are and what information, if not the classification decisions, we can extract from these visualizations.

Our contributions. We provide a theoretical explanation for why GBP and DeconvNet generate more human-interpretable but less class-sensitive visualizations than saliency map. Specifically, our analysis reveals that GBP and DeconvNet are essentially doing *(partial) image recovery* instead of highlighting class-relevant pixels or visualizing the learned weights, which means in principle they are unrelated to the decision-making of neural networks. We also find that it is the *backward ReLU* introduced by either GBP or DeconvNet, together with the *local connections* in CNNs that results in crisp visualizations. In particular, we explain how DeconvNet also relies on the max-pooling to recover the input. Finally, we do extensive experiments to support our theory and further reveal more detailed properties of these backpropagation-based visualizations.

2. Backpropagation-based Visualizations

In this section, we first give formal definitions of backpropagation-based visualizations: saliency map, DeconvNet and GBP, and then compare their empirical behaviors.

2.1. Formal Definitions

The key difference of backpropagation-based methods is the way they propagate the output score back through the ReLU activations. As illustrated by Figure 1, we consider

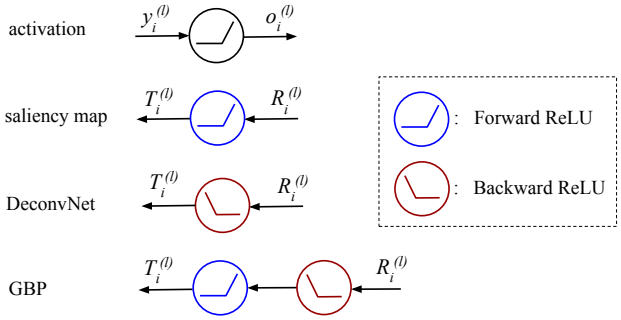


Figure 1. Illustration of how backpropagation-based methods propagate back through the i-th nonlinear activation in the l-th layer with input $y_i^{(l)}$ and output $o_i^{(l)}$, where $T_i^{(l)}$ denotes the (modified) gradient after passing through the activation and $R_i^{(l)}$ denotes the top gradient before the activation.

the i-th ReLU activation in the l-th layer with its input $y_i^{(l)}$ and its output $o_i^{(l)}$ and denote by $\sigma(t) = \max(t, 0)$ the ReLU activation. Also, denote by $R_i^{(l)}$ the top gradient before activation, i.e., gradient of the output score with respect to $o_i^{(l)}$ and denote by $T_i^{(l)}$ the (modified) gradient after activation, i.e., gradient of the output score with respect to $y_i^{(l)}$. Then in the gradient calculations, the corresponding *forward ReLU* could be formally defined as a function

$$\sigma_{f,i}^{(l)}(t) \triangleq \mathbb{I}\left(y_i^{(l)}\right) t$$

where $\mathbb{I}(\cdot)$ is the indicator function and the corresponding *backward ReLU* could be formally defined as a function

$$\sigma_{b,i}^{(l)}(t) \triangleq \mathbb{I}\left(R_i^{(l)}\right) t$$

Therefore, the formal definition of backpropagation-based methods for propagating the output score back through the i-th ReLU activation in the l-th layer is

$$T_i^{(l)} = \begin{cases} \sigma_{f,i}^{(l)}\left(R_i^{(l)}\right) & \text{for saliency map} \\ \sigma_{b,i}^{(l)}\left(R_i^{(l)}\right) & \text{for DeconvNet} \\ \sigma_{f,i}^{(l)}\left(\sigma_{b,i}^{(l)}\left(R_i^{(l)}\right)\right) & \text{for GBP} \end{cases}$$

which can be further uniformly formulated as

$$T_i^{(l)} = h\left(R_i^{(l)}\right) \frac{\partial g\left(y_i^{(l)}\right)}{\partial y_i^{(l)}} \tag{1}$$

where the two functions $h(\cdot)$ and $g(\cdot)$ are defined as

$$h(t) = \begin{cases} t & \text{for saliency map} \\ \sigma(t) & \text{for DeconvNet and GBP} \end{cases}$$
$$g(t) = \begin{cases} t & \text{for DeconvNet} \\ \sigma(t) & \text{for saliency map and GBP} \end{cases} \tag{2}$$

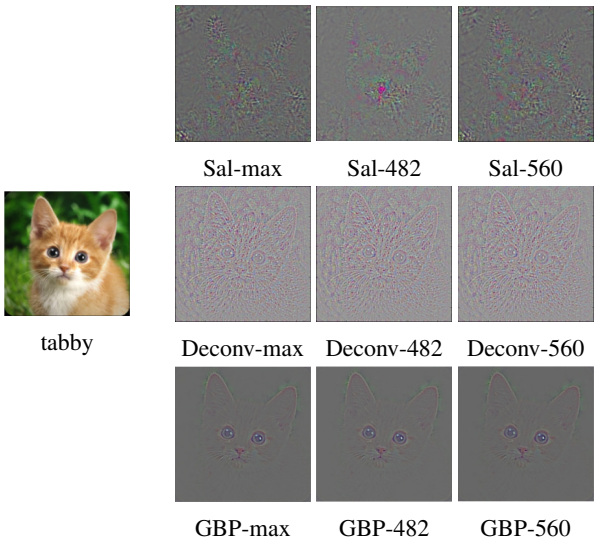


Figure 2. Backpropagation-based visualizations for the trained VGG-16 net given an input "tabby". From top row to the last row, it is saliency map, DeconvNet and GBP, where "max" refers to computing the (modified) gradient for the maximum class logit and the number, say "482", refers to computing the (modified) gradient for the 482-th logit. These numbers are randomly chosen for generality. Best viewed in the electronic version.

2.2. Empirical Observations

To be a good visualization method, a clean and visually human-interpretable result is very desirable. More importantly, it should also reveal how the neural networks make decisions. Based on this, we provide the empirical behaviors of the backpropagation-based visualizations for a pre-trained VGG-16 net (Simonyan & Zisserman, 2014) in Figure 2. Without loss of generality, the visualizations are obtained by choosing one of the class logits (i.e. the unnormalized class probability output right before the softmax function) as the output score to be taken derivative with respect to the input image.

For the visual quality, saliency map is very noisy while DeconvNet and GBP produce human-interpretable visualizations with a subtle difference: DeconvNet unexpectedly produces some kind of texture-like pattern, and GBP is cleaner with some background information filtered out. For the class-sensitivity, saliency map changes greatly for different class logits while DeconvNet and GBP are almost invariant to which class logit we choose. This, together with more experiments, suggests that after introducing the backward ReLU, both DeconvNet and GBP modify the true gradient in a way that they create much cleaner results but their functionality as an indicator of important pixels to a specific class has disappeared. In the next section, we will explain these empirical behaviors and discuss the reason why GBP and DeconvNet differ greatly from saliency map.

3. Theoretical Explanations

We first analyze the backpropagation-based methods in a three-layer CNN with random Gaussian weights, which is then extended to more complicated models such as CNNs with max-pooling and deep CNNs. Besides, we also investigate their behaviors in well-trained CNNs.

3.1. A Random Three-Layer CNN

Consider a three-layer CNN, consisting of an input layer and a convolutional hidden layer, followed by a ReLU activation function and a fully connected layer of which its output is called class logits. Formally, let $x \in \mathbb{R}^d$ be a normalized input image with dimension d and $\|x\| = 1$, and let $W \in \mathbb{R}^{p\times N}$ be N convolutional filters where each column $w^{(i)}$ denotes the i-th filter with size p. Note that here we use vectors to represent images and filters for simplicity, and the analysis also works for the more practical two-dimensional case. Then, we let $Y \in \mathbb{R}^{p\times J}$ be J image patches extracted from x, and each column $y^{(j)}$ with size p is generated by a linear function $y^{(j)} = D_j x$ where $D_j \triangleq \begin{bmatrix} 0_{p\times(j-1)b} & I_{p\times p} & 0_{p\times(d-(j-1)b-p)} \end{bmatrix}$ with b being the stride size[1]. For example, given a filter with size 3 and stride 1, the resulting j-th patch $y^{(j)}$ is made of the j-th to $(j+2)$-th consecutive pixels. The weights in the fully-connected layer can be represented by $V \in \mathbb{R}^{NJ\times K}$ with K being the number of output logits. Therefore, the k-th logit is represented by

$$f_k(x) = \sum_{i=1}^{N}\sum_{j=1}^{J} V_{q_{ij},k}\sigma(w^{(i)T}y^{(j)}) \tag{3}$$

where the index q_{ij} denotes the $((i-1)J+j)$-th entry in every column vector of weight matrix V.

Assume every entry of V and W is sampled from an *i.i.d.* Gaussian distribution $\mathcal{N}(0, c^2)$. The following lemma provides the formula for backpropagation-based visualizations in a random three-layer CNN. Note that the norm of the final results will be in the range of $[0, 1]$ as we apply the normalization during visualizations.

Lemma 1. *The backpropagation-based visualizations for the k-th logit in a random three-layer CNN is formalized as*

$$s_k(x) = \frac{1}{Z_k}\sum_{j=1}^{J} {D_j}^T \sum_{i=1}^{N} h(V_{q_{ij},k})\tilde{w}^{(i,j)} \tag{4}$$

where Z_k is the normalization coefficient to ensure $\|s_k(x)\| \in [0, 1]$, $h(\cdot)$ is given by Eq. (2) and

$$\tilde{w}^{(i,j)} = \begin{cases} w^{(i)} & \textit{for DeconvNet} \\ w^{(i)}\mathbb{I}\left(w^{(i)T}y^{(j)}\right) & \textit{for saliency map and GBP} \end{cases}$$

[1]Here we assume a VALID padding method implicitly, and other padding methods do not impact our analysis.

Proof. See Appendix A. □

Next, we can analyze the different behaviors of these backpropagation-based methods case by case.

3.1.1. GUIDED BACKPROPAGATION

First, the behavior of GBP is given as follows.

Theorem 1. *In a random three-layer CNN, if the number of filters N is sufficiently large, GBP at the k-th logit can be approximated as*

$$s_k^{GBP}(x) \approx x \tag{5}$$

Proof. See Appendix B. □

The above theorem shows that after introducing the backward ReLU, the input image can be *approximately recovered* by GBP in a random three-layer CNN, regardless of the class label. However, according to the linear model explanation, backpropagation-based methods are visualizing learned weights, which should be random noise as they are all sampled from *i.i.d* Gaussians. Obviously, it is inconsistent with the actual behavior of GBP.

As the approximation in Eq. (5) builds on an assumption that the number of filters N is sufficiently large, a key question is: How many filters are needed to guarantee an accurate recovery? From (Lugosi & Mendelson, 2017), we can set $N = \tilde{O}(\frac{p}{\epsilon^2})$ such that with high probability $\|\frac{1}{N}\sum_{i=1}^{N}\tilde{w}^{(i,j)} - \mathbb{E}[\tilde{w}^{(i,j)}]\| < \epsilon$, where p denotes the filter size and $\tilde{O}(\cdot)$ hides some other factors. As an upper bound, it reveals that the number of convolutional filters needed heavily depends on the filter size p. As the filter size intrinsically determined by the local connections in CNNs is usually small, we could use a mild number of convolutional filters to recover the input image. For example, given a filter size $3 \times 3 \times 3$, we need at most $O(10^3)$ filters to achieve an estimation error ϵ less than 0.1. This strongly suggests that GBP visualizations are human-interpretable in most of the CNNs, and thus the *local connections* property is another key factor underlying crisp visualizations.

3.1.2. SALIENCY MAP AND DECONVNET

Here we show the behaviors of saliency map and DeconvNet in a random three-layer CNN are largely different from GBP.

Theorem 2. *In a random three-layer CNN, if the number of filters N is sufficiently large, saliency map and DeconvNet are approximated as Gaussian random variables satisfying*

$$s_k^{Sal}(x), s_k^{Deconv}(x) \sim \mathcal{N}(0, I)$$

Proof. See Appendix C. □

The above theorem shows that both saliency map and DeconvNet visualizations will yield *random noise*, conveying little information about the input image and class logits. For saliency map, it is easily understood since saliency map represents the true gradient of the class logit, which heavily depends on the weights. For DeconvNet, although its behavior appears similar to saliency map in this simplistic scenario, we will show later on that it behaves more similarly to GBP, in particular with the existence of max-pooling.

3.2. Extensions to More Realistic Models

In this section, we extend our analysis of a simple random three-layer CNN to other more realistic cases, including the max-pooling, deeper nets and trained weights.

3.2.1. CNNS WITH MAX-POOLING

If we add a max-pooling layer between the ReLU and the fully-connected layer, the k-th logit becomes

$$f_k(x) = \sum_{i=1}^{N}\sum_{j=1}^{J} V_{\tilde{q}_{ij},k}\,\delta(\sigma(w^{(i)T}y^{(j)}))$$

where $\delta(\cdot)$ denotes the max-pooling, which successively selects the maximum value in a fixed-size pooling window, and the new index $\tilde{q}_{ij}$ is the down-sampled version of q_{ij}. Then the backpropagation-based visualizations for the k-th logit can be formulated as

$$s_k(x) = \frac{1}{Z_k}\sum_{j=1}^{J} {D_j}^T \sum_{i=1}^{N} h(\delta'(a_{ij})V_{\tilde{q}_{ij},k})\tilde{w}^{(i,j)} \tag{6}$$

where $a_{ij} \triangleq \sigma(w^{(i)T}y^{(j)})$ is the output of each ReLU activation and $\delta'(a_{ij})$ denotes the derivative of $\delta(\cdot)$ evaluated at a_{ij}, which is

$$\delta'(a_{ij}) = \begin{cases} 1 & \text{if } a_{ij} \text{ is chosen by max-pooling} \\ 0 & \text{otherwise} \end{cases}$$

Since $a_{ij} \geq 0$ with equality holds for $w^{(i)T}y^{(j)} \leq 0$, given a proper pooling window size, it is highly possible that a_{ij} is chosen by the max-pooling if and only if $w^{(i)T}y^{(j)} > 0$. It means with high probability, Eq. (6) is approximated as

$$s_k(x) \approx \frac{1}{Z_k}\sum_{j=1}^{J} {D_j}^T \sum_{i=1}^{N} h(V_{\tilde{q}_{ij},k})\tilde{w}^{(i,j)}\mathbb{I}(w^{(i)T}y^{(j)}) \tag{7}$$

For saliency map and GBP, we know $\tilde{w}^{(i,j)}\mathbb{I}(w^{(i)T}y^{(j)}) = \tilde{w}^{(i,j)}$ and thus Eq. (7) is further reduced to Eq. (4), which means the behaviors of saliency map and GBP remain the same after introducing the max-pooling. However, with high probability, DeconvNet at the k-th logit becomes

$$s_k^{\text{Deconv}}(x) \approx \frac{1}{Z_k}\sum_{j=1}^{J} {D_j}^T \sum_{i=1}^{N} \sigma(V_{\tilde{q}_{ij},k})w^{(i)}\mathbb{I}(w^{(i)T}y^{(j)})$$

which is exactly the form of GBP in Eq. (4). Therefore, adding the max-pooling makes the DeconvNet behave like GBP – doing nothing but image recovery. This also explains and extends the previous intuitive claims in (Samek et al., 2017; Odena et al., 2016) that the image-specific information in DeconvNet comes from the max-pooling.

Note that that the approximation from Eq. (6) to Eq. (7) in DeconvNet with the max-pooling is essentially different from the approximations used in GBP. For GBP, the approximate gap can be made arbitrarily small by increasing the hidden layer size N, leading to a perfect recovery of the input. However, for DeconvNet, given any pooling window size, there might always exist at least one of the following two contradictory cases: it is possible that a_{ij} is chosen by the max-pooling if $w^{(i)T}y^{(j)} \leq 0$, and also possible that a_{ij} is not chosen if $w^{(i)T}y^{(j)} > 0$. This makes DeconvNet (with max-pooling), in theory, never recover input perfectly, which might explain why the unusual texture-like artifacts appear in the DeconvNet visualizations.

3.2.2. Deep CNNs

The analysis for a three-layer CNN can be generalized to the multi-layer (or deeper) case. For clarity, we formulate the k-th logit of an L-layer deep CNN in a matrix form:

$$f_k(x) = \Gamma_k^{(L)T}\sigma\left(\Gamma^{(L-1)T}\cdots\sigma\left(\Gamma^{(1)T}x\right)\right)$$

where $\Gamma^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}}$ denotes either the convolutional or fully-connected operator matrix in the l-th layer and $\Gamma_k^{(L)}$ is the k-th column of $\Gamma^{(L)}$. Denote by $o^{(l)}$ the output of ReLU activations in the l-th layer, i.e. $o^{(l)} = \sigma\left(\Gamma^{(l)T}o^{(l-1)}\right),\ \forall l \in \{1,\cdots,L-1\}$ with $o^{(0)} \triangleq x$. Then backpropagation-based visualizations at the k-th logit in an L-layer deep CNN can be formulated as

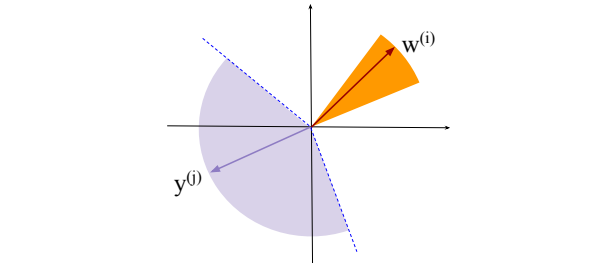
$$\begin{aligned} s_k(x) &= \frac{1}{Z_k}\frac{\partial\tilde{o}^{(1)}}{\partial x}\cdot h(\hat{V}_{\cdot,k}^{(1)}) \\ &\overset{(a)}{=} \frac{1}{Z_k}\sum_{j=1}^{J} {D_j}^T \sum_{i=1}^{N} h(\hat{V}_{q_{ij},k}^{(1)})\tilde{w}^{(i,j)} \end{aligned} \tag{8}$$

with $\forall l \in \{1,\cdots,L-1\}$,

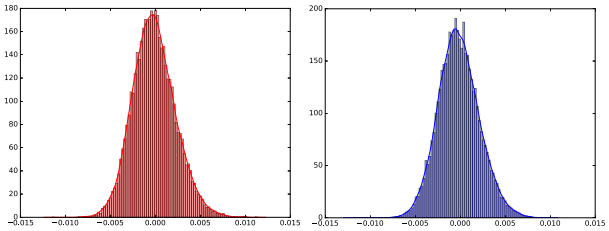
$$\hat{V}_{\cdot,k}^{(l)} = \frac{\partial\tilde{o}^{(l+1)}}{\partial o^{(l)}}\cdot h\left(\frac{\partial\tilde{o}^{(l+2)}}{\partial o^{(l+1)}}\cdots h\left(\frac{\partial\tilde{o}^{(L-1)}}{\partial o^{(L-2)}}h\left(\Gamma_k^{(L)}\right)\right)\right)$$

where in (a) we rewrite $s_k(x)$ in an expanded form, $\tilde{o}^{(l)} \triangleq g\left(\Gamma^{(l)T}o^{(l-1)}\right)$, $w^{(i)}$ is the i-th filter encoded in $\Gamma^{(1)}$ and N is the number of filters in the first convolutional layer. Also, $h(\cdot)$, $g(\cdot)$ and $\tilde{w}^{(i,j)}$ are defined in Eq. (2) and Lemma 1.

First, the approximate property of $\hat{V}_{\cdot,k}^{(1)}$ in the random deep CNN is given in the following proposition.



(a) A toy example



(b) neuron-26 in trained fc1 (c) neuron-44 in trained fc1

Figure 3. (a) shows a two-dimensional toy example where $w^{(i)}$'s are all in a cone (the orange area) and all the $y^{(j)}$'s in another cone (the grey area) called "dead zone" will be filtered out by the ReLU. (b) and (c) show the histograms of all weights connected to the 26-th activation and the 44-th one, respectively, in the layer "fc1" of the trained VGG-16 net. Note that we randomly picked up two activations (i.e. 26 and 44 here) for comparison.

Proposition 1. *For a random deep CNN where weights are i.i.d. Gaussians with zero mean, we can also approximate every entry of $\hat{V}_{\cdot,k}^{(1)}$ as i.i.d. Gaussian with zero mean.*

Proof. See Appendix D. □

Based on Proposition 1, we can see that the statistical properties of $\hat{V}_{q_{ij},k}^{(1)}$ in Eq. (8) are approximately the same with those of $V_{q_{ij},k}$ in Eq. (4), which means the analysis of backpropagation-based visualizations in a shallow three-layer CNN also applies to the deep CNN case. Therefore, the behaviors of these visualizations will barely change when increasing the depth of neural networks.

3.2.3. CNNs with Trained Weights

The previous analysis for random CNNs does not apply to the trained case directly since the weights here may not be *i.i.d.* Gaussian distributed. For saliency map, which uses the true gradient, the trained weights are likely to impose a stronger bias towards some specific subset of the input pixels, and so they can highlight class-relevant pixels rather than producing random noise. For GBP and DeconvNet, the analysis is a little more involved.

On the one hand, the trained weights $w^{(i)}$ will only lie in a small subspace of the whole image patch space which will create some "dead zones", as illustrated in Figure 3 (a). That

is, all image patches lying in the "dead zone" will be filtered out by the forward ReLU. For example, it is well-known that the trained weights in the first convolutional layer are Gabor-like filters to detect the image patches containing edges (Yosinski et al., 2014; Zeiler & Fergus, 2014). That is, image patches without edges will probably be filtered out by the first convolutional layer. Also, the higher convolutional layers keep filtering out more image patches with certain patterns (e.g. Figure 9). See the supplementary material for a comparison between GBP and a linear edge detector.

On the other hand, as shown in Figure 3 (b) and (c), the histograms of weights connected to the respective one of any two different neurons in the first fully connected layer (called "fc1") of the trained VGG-16 net are very similar to each other. Approximately, they form two very similar Gaussians with a small standard deviation, which means the (modified) gradients at any two different neurons in the layer "fc1" with respect to the input image are almost the same. Namely, $\frac{\partial \tilde{o}^{(\text{fc1})}}{\partial x}$ in Eq. (8) for GBP and DeconvNet (with max-pooling) satisfies

$$\frac{\partial \tilde{o}_m^{(\text{fc1})}}{\partial x} \approx F_{\text{conv}}(x), \forall m \in \{1, \cdots, M\}$$

where $\tilde{o}_m^{(\text{fc1})}$ is the m-th entry of $\tilde{o}^{(\text{fc1})}$ and $F_{\text{conv}}(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$ denotes the (normalized) overall filtering effect of the convolutional layers and M is the number of neurons in the layer "fc1". Thus, Eq. (8) for GBP and DeconvNet (with max-pooling) in the trained CNN can be approximated as

$$\begin{aligned} s_k(x) &= \frac{1}{Z_k} \frac{\partial \tilde{o}^{(\text{fc1})}}{\partial x} \cdot h(\hat{V}_{\cdot,k}^{(\text{fc1})}) \\ &= \frac{1}{Z_k} \sum_{m=1}^{M} \frac{\partial \tilde{o}_m^{(\text{fc1})}}{\partial x} \cdot h(\hat{V}_{m,k}^{(\text{fc1})}) \\ &\overset{(a)}{\approx} F_{\text{conv}}(x) \end{aligned} \tag{9}$$

where (a) follows from setting the normalization coefficient to be $Z_k = \frac{1}{\sum_{m=1}^{M} h(\hat{V}_{m,k}^{(\text{fc1})})}$.

It shows that GBP and DeconvNet (with max-pooling) in a trained CNN are actually doing the partial image recovery, where the trained weights control which image patch could form an *active path* to the class logit. More importantly, this filtering process is not class sensitive (e.g. the edge detector). In the end, only these "active" image patches are combined in the first fully connected layer to form the final visualization results. As the right side of (9) does not depend on k, it illustrates why the GBP and DecovNet visualizations in the trained VGG are not class-sensitive.

4. Experiments

To verify our theoretical analysis, we conduct a series of experiments on a three-layer CNN, a three-layer fully-connected network (FCN) and a VGG-16 net. For a random network, their weights are all sampled from the truncated Gaussians with a zero-mean and standard deviation 0.1. Unless stated otherwise, the input is the image "tabby" from the ImageNet dataset (Deng et al., 2009) with size $224 \times 224 \times 3$. See the supplementary materials for more results on other images and other neural network such as ResNet (He et al., 2016). In the three-layer CNN, the filter size is $7 \times 7 \times 3$, the number of filters is $N = 256$, and the stride is 2. In the three-layer FCN, the hidden layer size is set to $N_h = 4096$. By default, the backpropagation-based visualizations are calculated with respect to the maximum class logit.

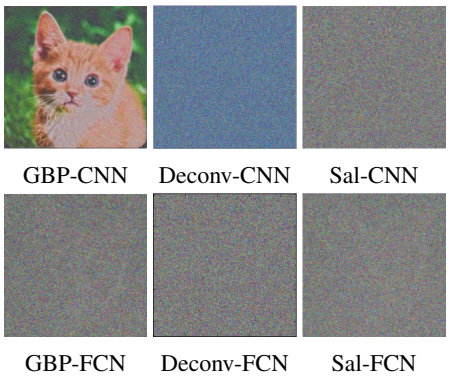


Figure 4. Backpropagation-based visualizations in a random three-layer CNN (top row) and a random three-layer FCN (bottom row) given the input image "tabby". From left to right, each column represents GBP, DeconvNet and saliency map, respectively. Only GBP visualization in the CNN is human-interpretable.

4.1. Impact of Local Connections

Figure 4 shows the backpropagation-based visualizations on a random three-layer CNN and a random three-layer FCN, respectively. We can see only GBP in the CNN can produce a human-interpretable visualization, while DeconvNet and saliency map in the CNN get random noise, which verifies our theoretical analysis in the section 3.1. In contrast, as local connections do not exist in the FCN and the input size (e.g. $224 \times 224 \times 3$) is extremely large, all the backpropagation-based methods (including GBP) in the FCN generate random noise. Particularly for GBP, the number of hidden neurons $N_h = 4096$ is still not large enough to recover the image.

To further highlight the impact of local connections in the visual quality of GBP, we vary the number of filters N in the CNN and the number of hidden neurons N_h in the FCN, respectively, while keep other parameters fixed. The results are given in Figure 5. Note that in the FCN, we have downsampled the input image to be of size $64 \times 64 \times 3$ due to computational limitations. We can see that as the number of filters N increases (resp. the hidden layer size N_h), the vi-

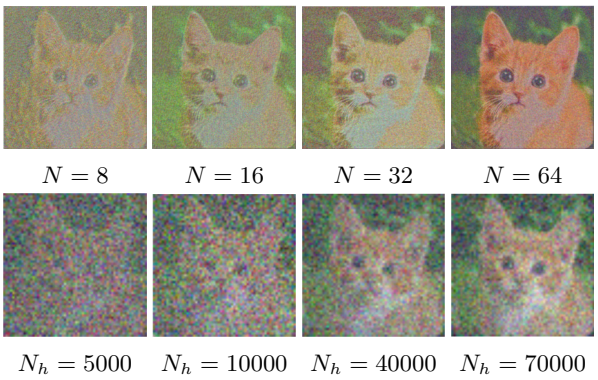


Figure 5. GBP visualizations given the input image "tabby" in a three-layer CNN (top row) by varying the number of filters N and in a three-layer FCN (bottom row) by varying the number of hidden neurons N_h.

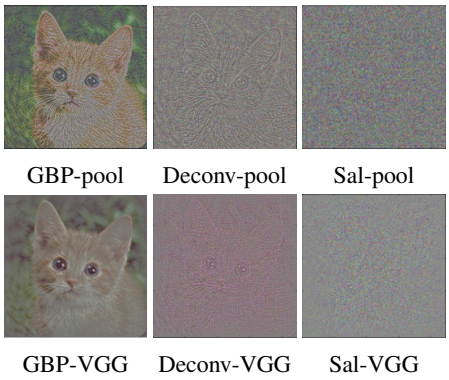


Figure 6. Backpropagation-based visualizations given the input image "tabby" in a random three layer CNN with the max-pooling (top row) and in a random VGG-16 net (bottom row). Now DeconvNet visualization also becomes human-interpretable.

sual quality of GBP in the CNN (resp. in the FCN) becomes better. Interestingly, even by setting $N_h = 70000$, which is definitely unrealistic, the FCN cannot achieve a comparable performance to the CNN with $N = 64$. Therefore, it confirms that the local connections in the CNN really contribute to the good visual quality of GBP.

4.2. Impact of Max-Pooling and Network Depth

To show the impact of the max-pooling in backpropagation-based visualizations, we then add a max-pooling layer in the above random three-layer CNN while keeping other parameters fixed, and the results are given in Figure 6 (top row). As compared with the visualizations in Figure 4 (top row), neither GBP or saliency map is impacted by the max-pooling, whereas the DeconvNet visualization has now become human interpretable instead of being the random noise as before. It confirms that the max-pooling is critical in helping DeconvNet produce human-interpretable visualizations via image recovery, as predicted by our theoretical analysis in the section 3.2.1.

To show the impact of network depth, we also apply backpropagation-based visualizations in a random VGG-16 net, which also includes the max-pooling but is much deeper than the three-layer CNN. Figure 6 (bottom row) shows that only saliency map generates random noise while both GBP and DeconvNet could produce human-interpretable visualizations. Though there are subtle visual differences between the top row and bottom row of Figure 6, the behaviors of backpropagation-based methods are basically unchanged after increasing the network depth. In addition, both GBP and DeconvNet reconstruct every fine-grained detail of the input image in the random VGG , which is different from the trained VGG in Figure 2 where only those "active" image patches are preserved.

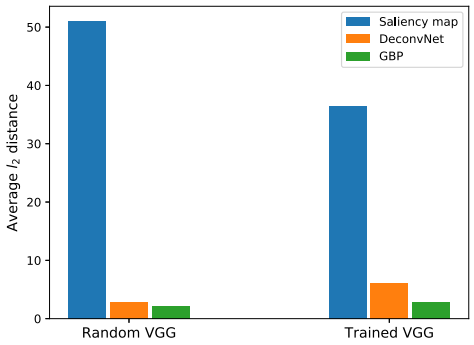


Figure 7. Average l_2 distance statistics. For each input, we randomly choose two class logits to get corresponding visualization results and calculate their l_2 distance. The above is the average l_2 distance based on 10K images from the ImageNet test set for each backpropagation-based method in both random and trained VGG-16 net.

4.3. Average l_2 Distance Statistics

To quantitatively describe how backpropagation-based visualizations change with respect to different class logits, we also provide the average l_2 distance statistics as shown in Figure 7. Our results are obtained by first calculating the l_2 distance of two visualization results given two different class logits for each input image and then taking an average of those l_2 distances based on 10K images from the ImageNet test set. The process is repeated for all backpropagation-based methods in both random and trained cases. As we can see, the average l_2 distance of saliency map is much larger than that of both GBP and DeconvNet in either a random VGG or a trained VGG, which clearly demonstrates that saliency map is class-sensitive but GBP and DeconvNet are not. Interestingly, in the trained VGG-16 net, the average l_2 distance of DeconvNet is slightly larger than that of GBP. It shows that the class insensitivity is exchanged for further

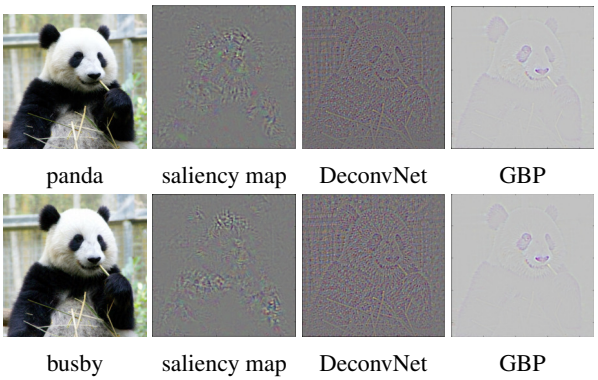


Figure 8. Top row: the image "panda" and its backpropagation-based visualizations. Bottom row: the adversarial example misclassified as "busby" and its backpropagation-based visualizations. Both experiments are applied in the trained VGG-16 net.

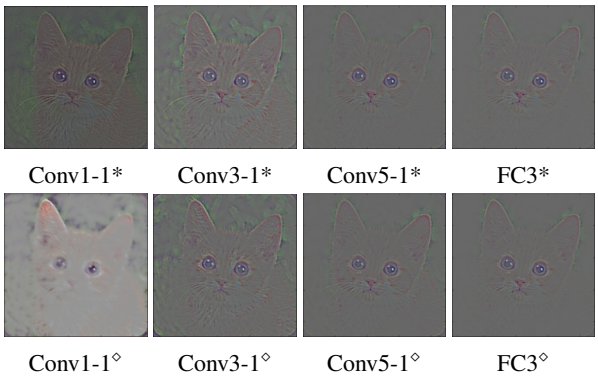


Figure 9. Top row: load trained weights **up to** the indexed layer and leave the later layers to be randomly initialized (marked by star sign). Bottom row: load trained weights **except for** the indexed layer is randomly initialized instead (marked by diamond sign).

improvement of visual quality.

4.4. Adversarial Attack on VGG

Adversarial attack provides another way of directly testing whether visualizations are class-sensitive or doing image recovery. The class-sensitive visualizations should change drastically as both the predicted class label and ReLU states of intermediate layers have changed, while the visualizations doing image recovery should change little as only a tiny adversarial perturbation is added into the input image. In this experiment, we first generate an adversarial example "busby" via the fast gradient sign method (FGSM) (Goodfellow et al., 2014) by feeding the image "panda" into the pre-trained VGG-16 net. Next, we apply the backpropagation-based visualizations to the original image "panda" and its adversary "busby" in the trained VGG-16 net. As shown in Figure 8, the saliency map visualization changes significantly whereas the GBP and DeconvNet visualizations remain almost unchanged after replacing "panda" by its adversary "busby". Therefore, it further confirms that saliency map is class-sensitive in that it highlights important pixels in making classification decisions. However, GBP and DeconvNet are doing nothing but (partial) image recovery.

4.5. VGG with Partly Trained Weights

There exist some differences for backpropagation-based visualizations, GBP and DeconvNet in particular, between the random and trained cases. We take GBP as an example here to investigate the contributions of different layers in the trained VGG-16 net to these visual differences.

First, to isolate the impact of later layers, we load the trained weights up to a given layer and leave later layers randomly initialized. As shown in Figure 9 (top row), from "Conv1-1*" to "Conv5-1*" GBP keeps filtering out more image patches as the number of trained convolutional layers increases. However, from "Conv5-1*" to "FC3*" (i.e., the fully-trained case) GBP behaves almost the same, no matter weights in the dense layers are random or trained. Therefore, it is the trained weights in the convolutional layers rather than those in the dense layers that account for filtering out image patches. Also, it further confirms that GBP is class-insensitive. Furthermore, to reveal the impact of each layer, we load the trained weights for the whole VGG-16 net except for a given layer which is randomly initialized instead. The results are shown in Figure 9 (bottom row). We can see that the GBP visualization is blurry for "Conv1-1◇", clean with much background information for "Conv3-1◇" and clean without background information for "Conv5-1◇", respectively. It means that the earlier convolutional layer has more important impact in the GBP visualization than the later convolutional layer.

5. Conclusions

In this paper, we proposed a theoretical explanation for backpropagation-based visualizations, where we started from a random three-layer CNN and later generalized it to more realistic cases. We showed that unlike saliency map, both GBP and DeconvNet are essentially doing (partial) image recovery, which verified their class-insensitive properties. We revealed that it is the backward ReLU, used by both GBP and DeconvNet, along with the local connections in CNNs, that is responsible for human-interpretable visualizations. We also explained how DeconvNet also relies on the max-pooling to recover the input. Our analysis was supported by extensive experiments. Finally, we hope our analysis can provide useful insights into developing better visualization methods for deep neural networks. A future direction is to understand how the GBP visualizations in the trained CNNs filter out image patches layer by layer.

Acknowledgements

Thanks to the anonymous reviewers for useful comments. WN, YZ and AB were supported by IARPA via DoI/IBC contract D16PC00003.

References

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255. IEEE, 2009.

Dosovitskiy, A. and Brox, T. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4829–4837, 2016.

Fong, R. C. and Vedaldi, A. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3429–3437, 2017.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Gunning, D. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Johnson, J., Alahi, A., and Fei-Fei, L. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pp. 694–711. Springer, 2016.

Kindermans, P.-J., Schütt, K. T., Alber, M., Müller, K.-R., and Dähne, S. Patternnet and patternlrp–improving the interpretability of neural networks. *arXiv preprint arXiv:1705.05598*, 2017.

Kraus, O. Z., Ba, J. L., and Frey, B. J. Classifying and segmenting microscopy images with deep multiple instance learning. *Bioinformatics*, 32(12):i52–i59, 2016.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Lugosi, G. and Mendelson, S. Sub-gaussian estimators of the mean of a random vector. *arXiv preprint arXiv:1702.00482*, 2017.

Mahendran, A. and Vedaldi, A. Salient deconvolutional networks. In *European Conference on Computer Vision*, pp. 120–135. Springer, 2016.

Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., and Clune, J. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in Neural Information Processing Systems*, pp. 3387–3395, 2016.

Odena, A., Dumoulin, V., and Olah, C. Deconvolution and checkerboard artifacts. *Distill*, 2016. doi: 10.23915/distill.00003. URL http://distill.pub/2016/deconv-checkerboard.

Samek, W., Binder, A., Montavon, G., Lapuschkin, S., and Müller, K.-R. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673, 2017.

Selvaraju, R. R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., and Batra, D. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *arXiv preprint arXiv:1610.02391*, 2016.

Shrikumar, A., Greenside, P., and Kundaje, A. Learning important features through propagating activation differences. *arXiv preprint arXiv:1704.02685*, 2017.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pp. 3320–3328, 2014.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.