DELVING DEEP INTO INTERPRETING NEURAL NETS WITH PIECE-WISE AFFINE REPRESENTATION

Yifu Chen, Antoine Saporta, Arnaud Dapogny and Matthieu Cord

Sorbonne University, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu, 75005 Paris

ABSTRACT

Deep convolutional neural networks (CNNs) are now ubiquitous in computer vision problems. However, these models usually describe very complicated functions of the input images. For a number of application, it is of utmost importance to be able to explain the decisions of a network, e.g. by highlighting the most relevant pixels in an image or a feature map w.r.t. a particular class. In this paper, we show that CNNs locally describe piece-wise affine functions of each pixel, whose coefficient and bias can be retrieved analytically. We apply our methodology on several popular CNNs and draw interesting conclusions on the relative contributions of pixels and biases for these networks.

Index Terms— Deep learning, deep neural networks, attribution, pixel contribution, bias

1. INTRODUCTION AND RELATED WORK

Nowadays deep convolutional neural networks (CNNs) are ubiquitous in computer vision. However, the lack of understanding behind these models makes them hard to interpret: it is however important to know why a CNN predicts what it does in order to trust the outcome of a deep learning-based intelligent system. Ideally, that means that a CNN should justify its prediction by identifying relevant parts in an image.

A number of approaches try to solve this problem: one particular way to deal with this [1] is to occlude a part of the image and see if observe a drop in accuracy. However, this method may need a lot of computation if we do not have any prior knowledge of the localization of the object. Another approach consists in estimating the contribution of each individual pixel (or features at different position) to the final (e.g. classification) task and then find the most discriminative parts. Nevertheless, there is no consensus on how to measure whether a pixel is discriminative or not. To this end, a popular solution is to consider the partial derivation of the classification score w.r.t the pixel as the importance of this pixel. An example of this method is guided back-propagation [2], which is however approximate as it essentially discards

This work was supported by the French National Agency (ANR) in the frame of its Technological Research (DS0705) 2016 program (Deep in France, project number ANR-13-CORD-0004).

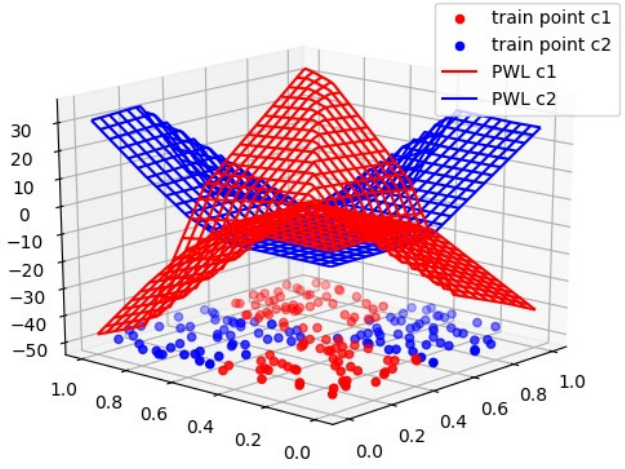

Fig. 1. Visualization of piece-wise affine representation of a simple neural network on a toy dataset of 2D generated points.

layer-wise negative contributions of the pixels. Authors in [3] uses the absolute value of the gradient at pixel position as a relevance score. Another caveheat of these methods is that it measures how a small modification of this pixel can affect the classification score, but not the contribution of this pixel for classification. For instance, if we consider a binary classifier $f(x_1, x_2) = x_1 x_2$ so that $x = (x_1, x_2) \in \mathbb{R}^+ \times \mathbb{R}^+$ belongs to C_1 if $f(x) > 1$. For an input $x = (0.1, 20)$, $\frac{\partial f}{\partial x_1} = 20 > 0.1 = \frac{\partial f}{\partial x_2}$. In such a case, if we define the gradient as the importance, then x_1 is way more important than x_2. However, intuitively $x_2 = 20$ is the main reason that $(0.1, 20)$ is classed to C_1 and $x_1 = 0.1$ actually has a negative evidence for class C_1. This simple example shows that *both* the gradient and the values shall be used, which is a premise of our work.

CAM [4] is a visualization method that is specific to architectures where the last convolution layer is followed by a global average pooling and then a fully connected layer, with no bias. If we note $f_k(i, j)$ the activation of unit k in the last convolution layer at spatial location (i, j), then the activation after global average pooling is $F^k = \sum_{i,j} f_k(i, j)$. For a given class c, if we write w_k^c the weight corresponding to class c for unit k, then the classification score of class c is $S_c =$

$\sum_k w_k^c F_k = \sum_{i,j} \sum_k w_k^c f_k(i,j)$. In this case, it is thus possible to define the importance of the activation of pixel i,j in feature map k for predicting class c as $\sum_k w_k^c f_k(i,j)$. However, this approach can only be used for specific architectures and lacks genericity to describe many popular deep networks.

Grad-CAM [5] generalizes this notion to all kind of CNNs. In this approach, the authors consider the coefficient w_k^c as the importance of feature map k for predicting class c. Since for an arbitrary CNN, the above formulation does not hold, the authors propose an approximation of w_k^c by averaging partial derivations of the classification score for class c w.r.t the activation of $f_k(i,j)$ as $\alpha_k^c = \frac{1}{Z} \sum_{i,j} \frac{\partial S^c}{\partial f_k(i,j)}$ where Z is the sum of activations of feature map k. However, this approximation considers only the 'importance' of different feature maps but ignores the spatial information inside the feature map. This can lead to a dire approximation, particularly when one wants to visualize the activations of upstream feature maps. In this paper, we propose a method to compute the exact contribution of each pixel of an image or an intermediate feature map that holds for every ReLU-based CNN. The rationale behind our method is that ReLU-based deep networks are locally equivalent to an affine function. Figure 1 illustrates this function obtained for a very simple ReLU-based network with 3 hidden layers with 5 units each. We apply our methods to several popular networks, showing its interest for visualization as well as understanding the network behavior. The contributions of this work are thus:

- We propose a new piece-wise affine representation of deep neural networks for distinguishing pixel-wise contributions in any feature map of a deep neural network. Our formulation is mathematically exact for ReLU-based deep networks.
- We apply our method for visualizing activations of popular networks such as VGG-19 and ResNet-101. We show both qualitatively and quantitatively that our method outperforms GradCAM and empirically show how batch normalization can bring robustness w.r.t. small perturbations of the inputs.

2. DEEP NEURAL NETS WITH RELU ACTIVATION ARE PIECE-WISE AFFINE FUNCTIONS

2.1. Structure of a feed-forward networks

In this paper, we focus on traditional feedforward deep neural networks with piece-wise linear activation such as ReLU[6]. ReLU ($g : x \to max(0,x)$) are widely used in modern networks, mainly because they offer good properties to avoid gradient vanishing while accelerating training process. For example VGG[7] and ResNet[8] all fall under this description. Consider a feed-forward deep neural network F. F can be expressed as a composition of elementary operations of its input X, *i.e.*:

$$F(X) = f_{out} \circ f_{L-1} \circ ... \circ f_1(X) \tag{1}$$

where f_l is usually either a linear transformation, ReLU, max/average pooling or batch normalization operator. All these operators are piece-wise affine functions. As a consequence, if we ignore the final softmax layer, a ReLU-based network is a piece-wise affine function of its inputs since any composition of piece-wise affine functions is still a piece-wise affine function. In other words, the input space is partitioned into a number of regions on which the network computes an affine function.

2.2. Classification score decomposition

Let's consider a ReLU-based neural network for a classification problem (the same consideration holds for a regression network). Given a input point $X \in \mathbb{R}^n$, we are going to explain how to define the contribution of each component x_i of X. Mathematically, the network defines a piece-wise affine function $F : \mathbb{R}^n \to \mathbb{R}^C$, where n is the input space dimension and C is the total number of classes. Several papers such as [9] have studied the upper bound of the number of linear regions. Since this number is finite and the intersection of linear regions is negligible, we can assume that X is not on the intersection of two linear regions. Therefore, X is located in the interior of a linear region $V(X)$ and the restriction of F to this region is an affine function $F_{|V(X)}$ defined as:

$$\begin{aligned} F_{|V(X)} : V(X) \subset \mathbb{R}^n &\to \mathbb{R}^C \\ Y &\mapsto W_{|V(X)} Y + b_{|V(X)} \end{aligned} \tag{2}$$

where $W_{|V(X)} = (w^c_{i|V(X)})_{c,i} \in \mathbb{R}^{C \times n}$ and $b_{|V(X)} = (b^c_{|V(X)})_c \in \mathbb{R}^C$ are constants. We note $F^c_{|V(X)}$ the c-th component of function F which represent the classification score of class c. According to Equation 2, $F^c_{|V(X)}$ is defined as :

$$\begin{aligned} F^c_{|V(X)} : V(X) \subset \mathbb{R}^n &\to \mathbb{R} \\ Y &\mapsto W^c_{|V(X)} Y + b^c_{|V(X)} \\ &= \sum_i w^c_{i|V(X)} y_i + b^c_{|V(X)} \end{aligned} \tag{3}$$

In this case, it makes sense to define the contribution of each variable. The contribution of variable x_i to the classification score $F^c_{|V(X)}$ is $S^c_{x_i} = w^c_{i|V(X)} x_i$. $S^c_{x_i} \geq 0$ implies that variable x_i has a positive contribution to class c while $S^c_{x_i} < 0$ means a negative contribution. Apart from the contributions of each variable, there is a bias term $b^c_{|V(X)}$ which define a 'prior knowledge' of all points in $V(X)$. The larger the bias is, the more robust the classification score will be w.r.t a small perturbation of X. Note at this point that $b^c_{|V(X)}$ is a constant that depends on $V(X)$, thus on the whole input X.

To sum it up, for any input X, the classification score of class c can be decomposed into contribution of each variable and a bias term. Moreover this decomposition is exact and unique due to the piece-wise affine property of the network.

2.3. Weights and bias computation

In this section, we show how to compute $W_{|V(X)}$ and $b_{|V(X)}$ efficiently for a given input X. Since for an affine function, the coefficient in front of x is equal to the gradient, $W^c_{|V(X)}$ can be easily calculated by a back-propagation:

$$W^c_{|V(X)} = \nabla F^c_{|V(X)}(X) \tag{4}$$

and $b^c_{|V(X)}$ can be obtained by:

$$b^c_{|V(X)} = F^c_{|V(X)}(X) - W^c_{|V(X)}X \tag{5}$$

Then, the contribution of the variable x_i to class c can be calculated simply by:

$$S^c(x_i) = w^c_{i|V(X)}x_i \tag{6}$$

Thus, we show that the importance of variable x_i is $w^c_{i|V(X)}x_i$ rather than the gradient $w^c_{i|V(X)}$. In other words, a pixel is important if both its value and gradient are large, which echoes the work of [10]. In the case of deep networks used to solve computer vision problems, X is an image and we can highlight the contribution of each of its pixels to F. In the upcoming section, we show the interest of this approach for visualization as well as understanding the network behavior. Also note that this decomposition can be applied at the level of any feature map within the network because of its compositional structure. (eg. $f_{out} \circ f_{L-1} \circ ... \circ f_l \circ f_{l-1} \circ ... \circ f_1$ is also a piece-wise affine function of $f_{l-1} \circ ... \circ f_1$)

3. EXPERIMENTS

In this Section, we show the interest of our method for visualization as well as a comparison with the popular GradCAM method, and use it to understand the behavior of several recent deep architectures, such as VGG-19 (5 blocks of 2 to 4 convolutional layers that will be denoted as conv1,... conv5 in what follows), VGG-19-BN (same network but with extra batch normalization layers) as well as ResNet-101.

3.1. Visualization and comparison with GradCAM

Figure 2 shows visualization of pixel-wise relevance of pixels from features maps at different levels of VGG-19 network, relatively to the final classification (class "cauliflower" for the two upper rows, and class "bull mastif" for the two lower rows). For both our approach and GradCam, negative contributions are discarded for visualization. For the last layers (conv5), GradCAM and our method outputs similar results. But for lower layers (conv4 and below), GradCAM is not precise as it involves spatial averaging of the gradients, which leads to coarse approximations. By contrast, our method is able to output more precise results, as the relevant regions are concentrated on the interesting objects in both cases.

In order to quantify this difference, we conducted an experiment on 100 randomly drawn images from Pascal VOC dataset. For each image, we computed the pixel-wise relevance of each pixel from the conv4 layer of VGG-19 network (from which our method and GradCAM differ a lot) for both our approach and GradCAM. Then, we select the 30% most relevant pixels relatively to both methods and zero'ed out the rest of the images. Finally, we applied a forward pass through VGG-19 network and, for each image, measured the intersection of the top-5 most relevant classes outputted by VGG-19 between (a) the whole image and the image masked with the relevance scores provided by our method, and (b) the whole image and the image masked with the relevance scores provided by GradCAM. This way, we measured the fidelity of the prediction outputted using only the most relevant information in the image, as highlighted by (a) our method, and (b) by GradCAM. We observed that in the first case, the top-5 intersection score was 35% vs 17% for GradCAM. This illustrates that our method more precisely highlights pixel-wise contributions to the classification score.

3.2. Bias as a robustness metric

We use our piece-wise affine decomposition of deep neural networks to investigate the importance of the bias term in the decomposition. Figure 3 shows the ratio between biases and the classification scores averaged among the 20 classes on 300 randomly sampled images from Pascal VOC 2012 validation set. We experiment with 3 popular architectures, namely VGG-19, VGG-19 with batch normalization and ResNet pretrained on ImageNet. First, we see that for the 3 networks, the bias is relatively important (from approximately 0.6 to 1 time the classification score) in the early stages of the network (first convolutional layers). It is far more preponderant for VGG-19-BN and ResNet-101 networks, indicating a greater robustness w.r.t. small perturbation of the input image or first feature maps. We also observe a steady decrease in bias importance for VGG-19 as we go upstream in the convolutional layers, while this decrease is far more abrupt for VGG-19-BN and ResNet towards the final classification layers. This means that batch normalization is a crucial element to put more emphasis on the bias *versus* the pixel-wise contribution. This, in turn, allows the networks to be more robust to small perturbation of its inputs or embeddings: this explains why, for instance, networks trained with batch normalization are more robust to e.g. adversarial attacks, at remarked in [11].

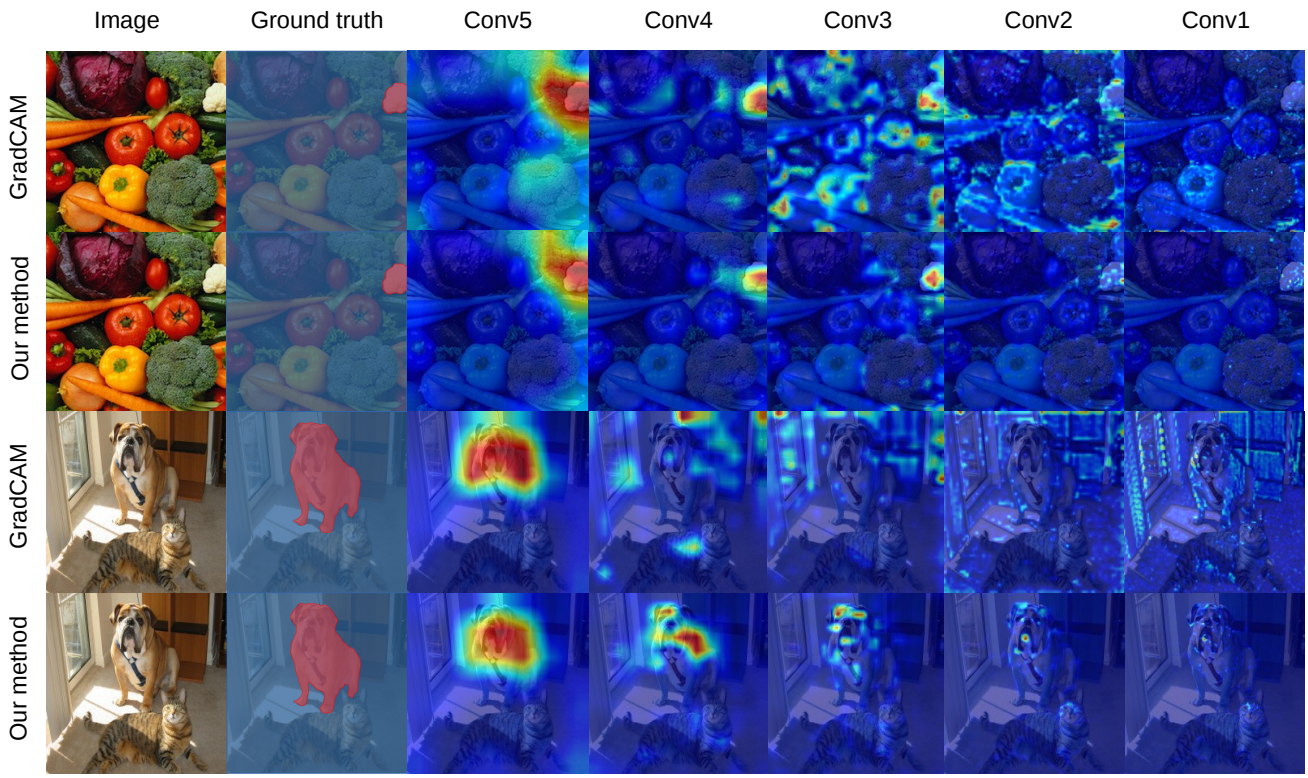

Fig. 2. Visualization of our method on two held-out images and VGG-19 network, and comparison with gradCAM. The deeper we go in the convolutional layers (Conv1 being the deepest), the more spread out the pixels highlighted by GradCAM are. Our method, however, is more precise as pixels belonging to the objects are relevant to the classification score.

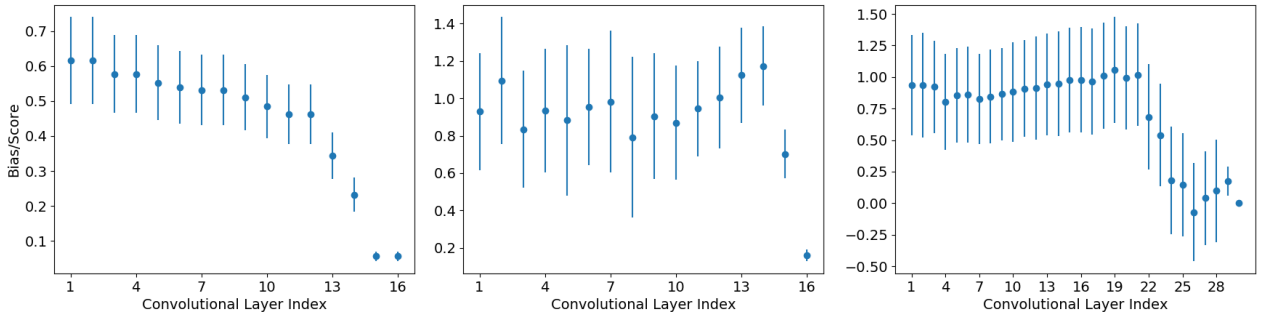

Fig. 3. Bias/score ratio as a function of the layer depth for popular deep architectures. Left to right: VGG-19, VGG-19 (with batch normalization), and ResNet-101.

4. DISCUSSION AND CONCLUSION

In this paper, we showed that ReLU-based deep neural networks are locally equivalent to piece-wise affine functions of its input. Given an image, this function can be calculated explicitly through gradient computation, providing a pixel-wise contribution to the classification score, as well as a global bias term. We show the interest of our method for vizalization of pixel-wise contribution at different levels of the network. Furthermore, we compare the pixel-wise contributions and biases for several networks, and show that deep networks trained with batch normalization have stronger biases, that decreases more abruptly towards the final classification layers. Thus batch normalization increase the robustness of the networks w.r.t. small perturbations of the images or the intermediate feature maps.

5. REFERENCES

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

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

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

[4] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba, "Learning deep features for discriminative localization," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[5] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu, "Interpretable convolutional neural networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8827–8836.

[6] Xavier Glorot, Antoine Bordes, and Yoshua Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. Springer, 2011, p. 315–323.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, 2015.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, , and Jian Sun, "Deep sparse rectifier neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[9] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio, "On the number of linear regions of deep neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[10] R. Fong and A. Vedaldi, "Interpretable explanations of black boxes by meaningful perturbation," in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[11] Nina Narodytska and Shiva Prasad Kasiviswanathan, "Simple black-box adversarial attacks on deep neural networks.," in *CVPR Workshops*, 2017, vol. 2.