Visual Explanation by Interpretation: Improving Visual Feedback Capabilities of Deep Neural Networks

Jose Oramas M.[*] **Kaili Wang**[*] **Tinne Tuytelaars**
KU Leuven, ESAT-PSI, IMEC

Abstract

Learning-based representations have become the de facto means to address computer vision tasks. Despite their massive adoption, the amount of work aiming at understanding the internal representations learned by these models is rather limited. Existing methods for model interpretation either require exhaustive manual inspection of visualizations, or link internal network activations with external "possibly useful" annotated concepts. In this paper, we propose an intermediate scheme in which, given a pretrained model, we automatically identify internal features relevant for the set of classes considered by the model, without requiring additional annotations. We *interpret* the model through average visualizations of these features. Then, at test time, we *explain* the network prediction by accompanying the predicted class label with supporting heatmap visualizations derived from the identified relevant features. In addition, we propose a method to address the artifacts introduced by strided operations in deconvnet-based visualizations. Moreover, we introduce an8Flower, a dataset specifically designed for the qualitative and quantitative evaluation of methods for model explanation. Our evaluation on the MNIST, ILSVRC12, Fashion144k and an8Flower datasets shows that the proposed method is able to identify relevant internal features for the classes of interest while improving the quality of the produced visualizations.

1 Introduction

In recent years, methods for learning-based representations based on deep neural networks (DNNs) have achieved impressive results in several computer vision tasks, e.g. image classification [4, 11, 14], object detection [20, 30], image generation [3, 8], etc. This, combined with the general tendency in the Computer Vision community of developing methods with a focus on high quantitative performance, has motivated the massive adoption of methods based on DNNs, which produce impressive quantitatively accurate predictions, despite their black-box characteristics. In this work, we aim for more visually-descriptive predictions and propose means to improve the quality of the visual feedback capabilities of DNN-based methods. Our goal is to bridge the gap between methods aiming at model *interpretation*, i.e. understanding what a given pre-trained model has actually learned, and methods aiming at model *explanation*, i.e. justifying the decisions made by a model.

Model interpretation of DNNs is commonly achieved in two ways: either by a) manually inspecting heatmap visualizations of every single filter (or a random subset thereof) from every layer of the network [28, 29] or, more recently, by b) exhaustively comparing the internal activations produced by a given model w.r.t. a dataset with pixel-wise annotations of possibly relevant concepts [2, 7]. These two paths have provided useful insights on the internal representations learned by DNNs. However, they have their own weaknesses. For the first case, the manual inspection of filter responses introduces a subjective bias, as was recently evidenced by [9]. In addition, the inspection of every filter from every layer becomes a cognitive-expensive practice for deeper models, which makes

[*]Denotes equal contribution

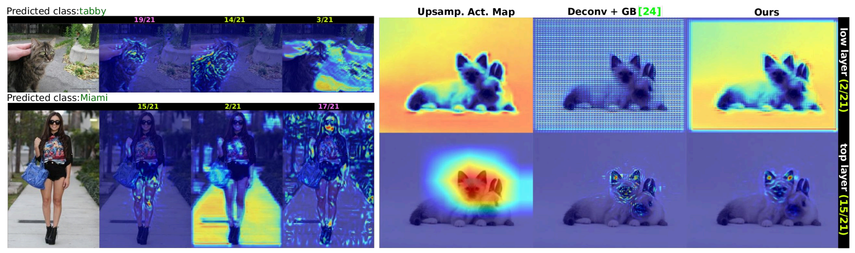

Figure 1: Left: Visual explanations generated by our method. Predicted class labels are enriched with heatmaps indicating the pixel locations, associated to the features, that contributed to the prediction. Note these features may come from the object itself as well as from its context. On top of each heatmap we indicate the number of the layer where the features come from. The layer type is color-coded (green for convolutional and pink for fully connected). Right: Visualization comparison. Note how our heatmaps attenuate the grid-like artifacts introduced by deconvnet-based methods at lower layers. Likewise, our method is able to produce a more detailed visual feedback than up-scaled activation maps.

it a noisy process. For the second case, as stated in [2], the interpretation capabilities over the network are limited by the concepts for which annotation is available. Moreover, the cost of adding annotations for new concepts is quite high due to its pixel-wise nature. A third weakness, that both cases share, is inherited by the way in which they generate spatial filter-wise responses, i.e. either through deconvolution-based heatmaps [23, 24, 29] or by up-scaling the activation maps at a given layer/filter to the image space [2, 32]. On the one hand, deconvolution-based methods are able to produce heatmaps with high level of detail from any filter in the network. However, as can be seen in Fig. 1 (right), they suffer from artifacts introduced by strided operations in the back-propagation process. On the other hand, up-scaled activation maps can significantly lose details when displaying the response of filters with large receptive field from deeper layers. Moreover, they have the weakness of only being computable for convolutional layers.

In order to alleviate these issues, we start from the hypothesis, empirically proven by previous work [2, 28], that a small subset of the internal filters of a network encode features that are important for the task that the network addresses. Based on that assumption, we propose a method which, given a trained DNN model, automatically identifies a set of relevant internal filters whose encoded features serve as indicators for the class of interest to be predicted. These filters can originate from any type of internal layer of the network, i.e. *convolutional, fully connected*, etc. This is formulated as a μ-*Lasso* optimization problem in which a sparse set of filter-wise responses are linearly combined in order to predict the class of interest. At test time, given an image, a set of identified relevant filters, and a class prediction, we accompany the predicted class label with heatmap visualizations of the top-responding relevant filters for the predicted class, see Fig. 1 (left). In addition, by improving the resampling operations within deconvnet-based methods [24, 29], we are able to address the artifacts introduced in the back-propagation process, see Fig. 1 (right). The code and models used to generate our visual explanations will be released upon the publication of this manuscript. Overall, the proposed method removes the requirement of additional expensive pixel-wise annotation, by relying on the same annotations used to train the initial model. Moreover, by using our own variant of a deconvolution-based method, our method is able to consider the spatial response from any filter at any layer while still providing visually pleasant feedback. This allows our method to reach some level of explanation by interpretation. Finally, recent approaches to evaluate explanation methods measure the quality of explanation by verifying its effect on a proxy task, e.g. classification [29] and object localization [33, 32]. While a good explanation should have a direct effect on the task of interest, it should also have a good coverage of the feature being explained. Here we propose *an8Flower*, a synthetic dataset where the discriminative feature between the classes of interest is controlled. This allows us to produce ground-truth masks for the regions to be highlighted as part of the explanation. At the same time it allows us to quantitatively measure the performance of methods for model explanation.

From a practical point of view, algorithms capable of communicating, as part of their output, the train of thought taken to reach a particular prediction are more likely to be trusted and adopted by end users than systems that operate in a black-box fashion. Moreover, findings and techniques developed in this direction may assist the overhaul required for current machine learning technology in order to meet recently approved legislation, i.e. EU General Data Protection Regulation (art. 13-15 & 22).

The main contributions of this work are four-fold. First, we propose an automatic non-exhaustive method based on relevant feature selection to identify the network-encoded features that are important for the prediction of a given class. This alleviates the requirement of exhaustive manual inspection or additional expensive pixel-wise annotations required by existing methods. Second, the proposed method is able to provide visual feedback with higher-level of detail over up-scaled raw activation maps [2, 32], and improved quality over recent deconvolution-based methods with guided back-propagation [24, 29]. Third, the proposed method is general enough to be applied to any type of network, independently of the type of layers that compose it. Fourth, we release *an8Flower*, a dataset and evaluation protocol specifically designed for evaluating the accuracy of methods for model explanation. To the best of our knowledge this is the first dataset aimed at such task.

This paper is organized as follows: in Section 2 we position our work w.r.t. existing work. Section 3 presents the pipeline and inner-workings of the proposed method. In Section 4, we conduct a series of experiments evaluating different aspects of the proposed method while discussing the observations and findings made throughout our evaluation. Finally, in Section 5, we conclude this paper.

2 Related Work

This work lies at the intersection between model interpretation and explanation for DNNs. These two groups of work constitute the axes along which we position our work.

Interpretation. Input modification methods [10, 29, 33] were one of the earlier attempts towards DNN interpretation in recent time. These methods see the network as a black-box. They are designed to visualize properties of the function this black-box represents by systematically covering (part of) the input image and measuring the difference of activations. Their assumption is that occlusion of important parts of the input will lead to a significant drop in performance. This procedure can be applied at test time to identify the regions of the image that are important for classification. However, the resolution of the explanation will depend on the region size, which will inversely increase computation cost. We apply the core idea made by this group of work in our evaluation in order to verify the relevance of the internal network features selected by our method.

A more recent group of works focuses on linking internal DNN activations with semantic concepts. [6] proposed a feature selection method in which the neuron activations of a DNN trained with object categories are linearly combined to predict object attribute classes. Following a similar idea, [2, 7, 27, 31] proposed to exhaustively match the internal activations of every filter from the convolutional layers against a dataset with pixel-wise annotated concepts. While both methods provide important insights on the semantic concepts encoded by the filters of the network, they are both limited by the concepts for which annotation is available. Similar to [6], we discover relevant internal filters through a feature selection method. Different from [6], we link internal network activations directly to the same annotations used to train the initial model. This effectively removes the expensive requirement of additional annotations.

A third line of works aims at discovering frequent mid-level visual patterns [5, 16, 19] occurring in image collections. The constraints enforced during the mining of these mid-level elements empowers them with high semantic coverage, making them effective for training appearance classifiers or as means for summarization. We adopt the idea of using visualizations of (internal) mid-level elements as means to reveal the relevant features encoded, internally, by a DNN. More precisely, we use the average visualizations used by these works in order to interpret, visually, what the network has actually learned.

Explanation. For sake of brevity, we ignore methods which generate explanations via bounding boxes [13, 18] or text [12], and focus on methods capable of generating visualizations with pixel-level precision. Works belonging to this group map the activity at a given layer/filter back to the input image space. [29] proposed a multi-layered deconvolutional network (Deconvnet) which uses activations from a given top layer and reverses the path of excitatory stimuli to reveal which visual patterns from the input image are responsible for the observed activations. In parallel, [23] proposed a variation of [29] where information from the lower layers and the input image is used to estimate which image regions are responsible for the activations seen at the top layers. In a similar direction, [1] aims at decomposing the classification decision into a pixel-wise contribution while enforcing a layer-wise conservation principle in which the propagated quantity should be preserved between neurons from adjacent layers. Later, [24] extended these works by introducing "guided back-propagation", a technique that removes the effect of units with negative activations during the forward pass and with negative contributions in the backwards pass. This resulted in improvements in sharpness and clarity of the generated visualizations. The main difference between [23, 24, 29] lies in the way in which the ReLU operation is handled during the backwards pass in order to introduce the desired effect. [32] proposes to ignore fully connected layers and uses a Global Average Pooling (GAP) operation at the end of the last convolutional layer. This GAP operation is a weighted sum over the spatial locations of the activations of the filters of the last convolutional layer, which results in a class activation map. Finally, a heatmap is generated by upsampling the class activation map to the size of the input image.

Here, we take the Deconvnet-based methods as starting point given their maturity and their ability to produce visual feedback with pixel-level precision. However, we change the internal operations in the backward pass with the goal of reducing visual artifacts introduced by strided operations while maintaining the network structure.

3 Proposed Method

The proposed method consists of two parts. At training time, a set of relevant layer/filter pairs are identified for every class of interest j. This results in a relevance weight w_j, associated to class j, for every filter-wise response x computed internally by the network. At test time, an image I is pushed through the network producing the class prediction $\hat{j}{=}F(I)$. Then, taking into account the internal responses x, and relevance weights $w_{\hat{j}}$ for the predicted class $\hat{j}$, we generate visualizations indicating the image regions that contributed to this prediction.

3.1 Identifying Relevant Features

One of the strengths of deep models is their ability to learn abstract concepts from simpler ones. That is, when an example is pushed into the model, a conclusion concerning a specific task can be reached as a function of the results (activations) of intermediate operations at different levels (layers) of the model. These intermediate results may hint at the "semantic" concepts that the model is taking into account when making a decision. From this observation, and the fact that activations are typically sparse, we make the assumption that some of the internal filters of a network encode features that are important for the task that the network addresses. To this end, we follow a procedure similar to [6], aiming to predict each class j by the linear combination $w_j{\in}\mathbb{R}^m$ of its internal activations x, with m the total number of neurons/activations.

As an initial step, we extract the image-wise response x_i. To this end, we compute the L_2 norm of each channel (filter response) within each layer and produce a 1-dimensional descriptor by concatenating the responses from the different channels. This layer-specific descriptor is L_1-normalized in order to compensate for the difference in length among different layers. Finally, we concatenate all the layer-specific descriptors to obtain x_i. Note that, in this process, we do not consider the last layer whose output is directly related to the classes of interest, e.g. the last two layers from VGG-F [4].

Following this procedure, we construct the matrix $X{\in}\mathbb{R}^{m\times N}$ by passing each of the N training images through the network F and collecting the internal responses x. As such, the i^{th} image of the dataset is represented by a vector $x_i{\in}\mathbb{R}^m$ defined by the filter-wise responses at different layers. Furthermore, the possible classes that the i^{th} image belongs to are organized in a binary vector $l_i{\in}\{0,1\}^C$ where C is the total number of classes of interest. Putting the annotations from all the images together produces the binary label matrix $L{=}[l_1, l_2, ..., l_N]$, with $L{\in}\mathbb{R}^{C\times N}$. With these terms in place, we resort to solving the equation:

$$W^* = argmin_W \; ||X^T W - L^T||_F^2 \quad subject\, to: \; ||w_j||_1 \leq \mu \; , \; \forall_j = 1, ..., C \tag{1}$$

with μ a parameter that allows controlling the sparsity. This is the matrix form of the μ-Lasso problem. This problem can be efficiently solved using the Spectral Gradient Projection method [17, 25]. After solving the μ-Lasso problem, we have a matrix $W{=}[w_1, w_2, ..., w_C]$, with $W{\in}\mathbb{R}^{m\times C}$. We impose sparsity on W by enforcing the constraints on the L_1 norm of w_j, i.e. $||w_j||_1{\leq}\mu$, $\forall_j = 1, ..., C$. As a result, each non-zero element in W represents a pair of network layer p and filter index q (within the layer) of relevance.

3.2 Generating Visual Feedback

During training time (Section 3.1), we identified a set of features with relevance weights W for the classes of interest. At test time, we generate the feedback visualizations by taking into account the response of the relevant features on the content of the tested images.

Towards this goal, we push an image I through the network producing the class prediction $\hat{j}{=}F(I)$. In parallel, we compute the internal filter-wise response vector x_i following the procedure presented

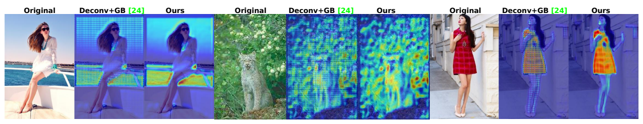

Figure 2: Heatmap visualization at lower layers of VGG-F . Note how our method attenuates the grid-like artifacts introduced by existing deconvnet-based (Deconv+GB) methods [24, 29].

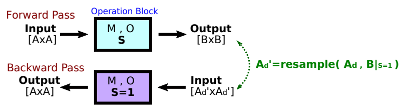

Figure 3: To attenuate artifacts produced by strided operations, during the backward pass, we set the stride to 1 ($S = 1$) and compensate by resampling the input so that $A'_d = B|_{S=1}$.

above. Then we compute the weighted response $r_i^{\hat{j}}{=}(w_{\hat{j}} \circ x_i)$, where $\circ$ represents the element-wise product between two vectors. The features, i.e. layer/filter pairs (p^*, q^*), with strongest contribution in the prediction $\hat{j}$ are selected as those with maximum response in $r_i^{\hat{j}}$. Finally, we feed this information to the Deconvnet-based method with guided backpropagation [24] from [10] to visualize the important features as defined by the layer/filter pairs (p^*, q^*). Following the visualization method from [10], given a filter p from layer q and an input image, we first push forward the input image through the network, storing the activations from each filter at each layer, until reaching the layer p. Then, we backpropagate the activations from filter q at layer p with inverse operations until reaching back to the input image space. As as result we get as part of the output a set of heatmaps, associated to the relevant features, defined by (p^*, q^*), indicating the relative influence of the pixels that contributed to the prediction. See Fig.1 (left) for an example of the visual feedback provided by our method. Please refer to [10, 24, 29] for further details regarding Deconvnet-based methods.

3.3 Improving Visual Feedback Quality

Deep neural networks addressing computer vision tasks commonly push the input visual data through a sequence of operations. A common trend of this sequential processing is that the input data is internally resampled until reaching the desired prediction space. As mentioned in Sec. 2, methods aiming at interpretation/explanation start from an internal point in the network and go backwards until reaching the input space - producing a heatmap. However, due to the resampling process, heatmaps generated by the backwards process tend to display grid-like artifacts. More precisely, we find that this grid effect is caused by the internal resampling introduced by network operations with stride larger than one ($S{>}1$). To alleviate this effect, in the backwards pass, we set the stride $S{=}1$ and compensate this change by modifying the input accordingly. As a result, the backwards process can be executed while maintaining the network structure.

More formally, given a network operation block defined by a convolution mask with size $[M{\times}M]$, stride $[S, S]$, and padding $[O, O, O, O]$, the relationship between the size of its input $[A{\times}A]$ and its output $[B{\times}B]$ (see Fig. 3) is characterized by the following equation:

$$A + 2 \cdot O = M + (B - 1) \cdot S \tag{2}$$

from where,

$$B = [\ (A + 2 \cdot O - M)/S\ ] + 1 \tag{3}$$

Our method starts from the input ($[A_d{\times}A_d]$), which encodes the contributions from the input image, carried by the higher layer in the Deconvnet backward pass. In order to enforce a "cleaner" resampling when $S{>}1$, during the backward pass, the size of the input ($[A_d{\times}A_d]$) of the operation block should be the same as that of the feature map ($[B{\times}B]$) produced by the forward pass if the stride S were equal to one, i.e. $A'_d{=}B|_{S=1}$. According to Eq. 3 with $S{=}1$, A_d should therefore be resampled to $A'_d{=}B|_{S=1}{=}A{+}2 \cdot O{-}M + 1$. We do this resampling via the nearest-neighbor interpolation algorithm given its proven fast computation time which makes it optimal for real-time processing. By introducing this step, the network will perform the backwards pass with stride $S{=}1$ and the grid effect will disappear. See Fig. 2 for some examples of the improvements introduced by our method.

4 Evaluation

We conduct three sets of experiments. In the first experiment (Sec. 4.1), we verify the importance of the identified relevant features in the task addressed by the network. In the second experiment (Sec. 4.2), we evaluate the improvements on visual quality provided by our method. Finally, in the third experiment (Sec. 4.3), we quantify the capability of our visual explanations to highlight the regions that are descriptive for the classes of interest.

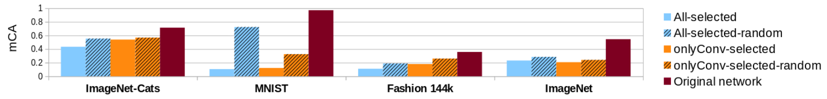

Figure 4: Changes in mean classification accuracy (mCA) as the identified relevant filters are ablated.

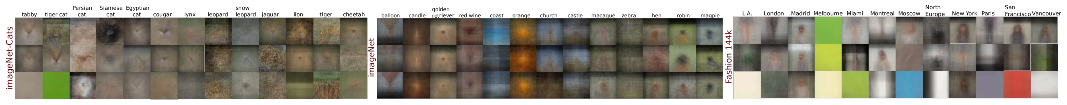

Figure 5: Average Images from the identified relevant filters for the ImageNet-Cats subset (left), some selected classes from the full ImageNet (center) and the Fashion144K (left) datasets, respectively.

Evaluation Protocol. We evaluate the proposed method on an image recognition task. We conduct experiments on three standard image recognition datasets, i.e. MNIST [15], Fashion144k [22] and imageNet (ILSVRC'12) [21]. Additionally, we conduct experiments on a subset of cat images from imageNet (imageNet-cats). MNIST covers 10 classes of hand-written digits. It is composed by 70k images in total (60k for training/validation, 10k for testing). The imageNet dataset is composed of 1k classes. Following the standard practice, we measure performance on its validation set. Each class contains 50 validation images. For the Fashion144k dataset [22], we consider the subset of 12k images from [26] used for the geolocation of 12 city classes. The imageNet-cats subset consists of 13 different cat classes, containing both domestic and wild cats. It is composed of 17,550 images. Each class contains 1,3k images for training and 50 images for testing. Please refer to the supplementary material for more implementation details.

4.1 Importance of Identified Relevant Features

In this experiment we verify the importance of the "relevant" features identified by our method at training time, see Sec. 3.1. To this end, given a set of identified features we evaluate the influence they have in the network by measuring changes in classification performance caused by their removal. We remove features in the network by setting their corresponding layer/filter to zero. The expected behavior is that a set of features with high relevance will produce a stronger drop in performance when ablated. In Fig. 4, we show the changes in classification performance for the tested datasets. We report the performance of four sets of features: a) *All*, selected with our method by considering the whole internal network architecture, b) *OnlyConv*, selected by considering only the convolutional layers of the network, c) a *Random* selection of features (filters) selected from the layers indicated in the sets a) and b), and for reference, d) the performance obtained by the original network. Note that the *OnlyConv* method, makes the assumption that relevant features are only present in the convolutional layers. This is a similar assumption as the one made by state-of-the-art methods [2, 27, 32]. When performing feature selection (Sec.3.1), we set the sparsity parameter μ=10 for all the tested datasets. This produces subsets of 92|101, 46|28, 104|111, 248|180 relevant features for the *All*|*OnlyConv* methods, on the respective datasets from Fig. 4. Differences in the number of the selected features can be attributed to possibly redundant or missing predictive information between the initial pools of filter responses x used to select the *All* and *OnlyConv* features.

A quick inspection of Fig. 4 shows that indeed classification performance drops when we remove the identified features, *All* and *OnlyConv*. Moreover, it is noticeable that a random removal of features has a lower effect on classification accuracy. This demonstrates the relevance of the identified features for the classes of interest. In addition, it is visible that the method that considers the complete internal structure, i.e. *All*, suffers a stronger drop in performance compared to the *OnlyConv* which only considers features produced by the convolutional layers. This suggests that there is indeed important information encoded in the fully connected layers, and while convolutional layers are a good source for features, focusing on them only does not reveal the full story.

But... does it make sense? In order to get a qualitative insight of the type of information that these features encode we compute an average visualization by considering the top 100 image patches where such features have a high response. Towards this goal, given the set of identified relevant features, for every class, we select images with higher responses. Then, we take the input image at the location with maximum response for a given filter and crop it by considering the receptive field of the corresponding layer/filter of interest. Selected examples of average images, with rich semantic representation, are presented in Fig. 5 for the full imageNet, the imageNet-Cats subset, and the Fashion144k datasets.

We can notice that for imageNet-Cats, the identified features cover descriptive characteristics of the considered cat classes. For example, the dark head of a siamese cat, the nose/mouth of a cougar,

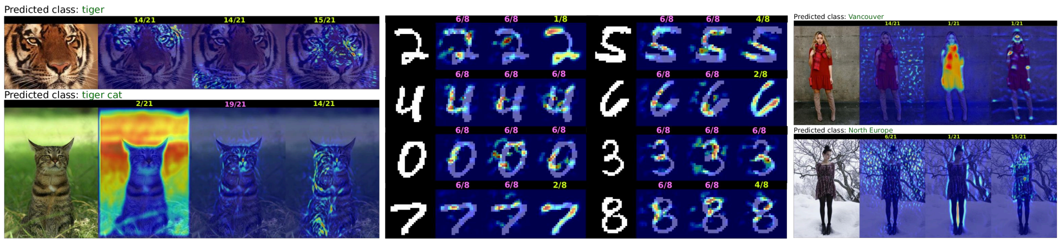

Figure 6: Our visual explanations. We accompany the predicted class label with our heatmaps indicating the pixel locations, associated to the features, that contributed to the prediction. Note that the features may come from the object itself as well as from its context. See how for the MNISTexamples, some features support the existence of gaps, as to avoid confusion with another class. On top of each heatmap we indicate the number of the layer where the features come from. The layer type is color-coded, i.e. convolutional (green) and fully connected (pink).

or the fluffy-white body shape of persian cat. Likewise, it effectively identifies the descriptive fur patterns from the jaguar, leopard and tiger classes and colors which are related to the background. We see a similar effect on a selection of other objects from the rest of the imageNet dataset. For instance, for scene-type classes, i.e. coast, castle and church, the identified features focus on the outline of such scenes. Similarly, we notice different viewpoints for animal-type classes, e.g. golden-retriever, hen, robin, magpie. On the Fashion144k dataset (Fig. 5 (right)) we can notice that some classes respond to features related to green, blue, red, and beige colors. Some focus on legs, covered and uncovered, while other focus on the upped body part. It is interesting that from the upper body parts, some focus on persons with dark long hair, short hair, and light hair. Similarly, there is a class with high response to horizontal black-white gradients where individuals tend to dress in dark clothes. These visualizations answers the question explored in [26] and why the computer outperforms the surveyed participants. It shows that the model effectively exploits human-related features (legs clothing, hair length/color, clothing color) as well as background-related features, mainly covered by color/gradients and texture patterns. In the visual explanations provided by our method we can see that the model effectively uses this type of features to reach its decision.

Finally, in Fig. 6 we show some examples of the visual explanations produced by our method. We aggregate the predicted class label with our heatmap visualizations indicating the pixel locations, associated to the relevant features, that contributed to the prediction. For the case of the ILSVRC'12 and Fashion144k examples, we notice that the relevant features come from the object itself as well as from its context. For the case of the MNISTexamples, in addition to the features firing on the object, there are features that support the existence of a gap (background), as to emphasize that the object is not filled there and avoid confusion with another class. For example, see for class 2 how it speaks against 0 and for 6 how it goes against 4.

4.2 Visual Feedback Quality

In this section we verify the visual quality of the visualizations generated by our method as part of the prediction feedback. Towards this goal, we compare our visualizations with upsampled activation maps from internal layers [2, 32] and the output of Deconvnet combined with guided-backpropagation [24]. In Fig. 7 we present examples showing the heatmap visualization for different methods. We show these visualization for different layers/filters throughout the network.

A quick inspection reveals that our method to attenuate the grid-like artifacts introduced by Deconvnet methods (see Sec 3.3) indeed produces noticeable improvements for lower layers. See Fig. 2 for additional examples presenting this difference at lower layers. Likewise, for the case of higher layers (Fig. 7), the proposed method provides more precise visualizations when compared to upsampled activation maps. In fact, the rough output produced by the activation maps at higher layers has a saliency-like behavior that gives the impression that the network is focusing on a larger region of the image. This could be a possible attribution to why in earlier works [33], manual inspection of network activations suggested that the network was focusing on "semantic" parts. Please see [9] for an in-depth discussion of this observation. Finally, for the case of FC layers, using upsampled activation maps is not applicable. Please refer to the supplementary material for additional examples. In addition, to quantitatively measure the quality of our heatmaps we perform a box-occlusion study [29]. Given a specific heatmap, we occlude the original image with patches sampled from the distribution defined by the heatmap. We measure changes in performance as we gradually increase the number of patches up to covering the 30% most relevant part of the image. Here our method reaches a mean difference

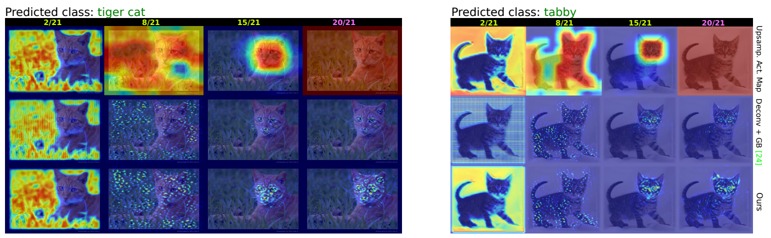

Figure 7: Pixel effect visualization for different methods. Note how for lower layers (8/21), our method attenuates the grid-like artifacts introduced by Deconvnet methods. Likewise, for higher layers (15/21), our method provides a more precise visualization when compared to upsampled activation maps. For the case of FC layers (20/21), using upsampled activation maps is not applicable.

	an8Flower	
Method	single-6c	double-12c
Upsam. Act.	0.15	0.19
Deconv+GB [24]	0.22	0.20
Ours	0.23	0.21

Figure 8: Area under the IoU curve. We report the performance of the proposed feature selection method using three different means to generate heatmap visualizations.

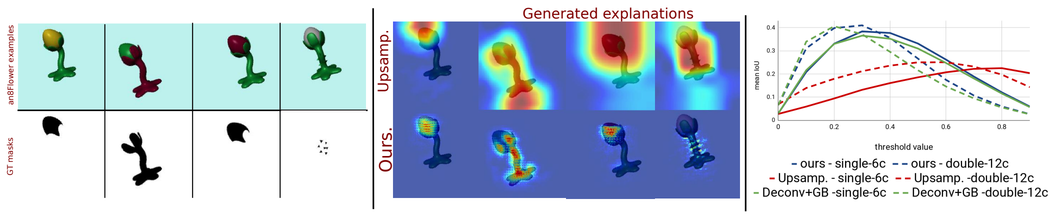

Figure 9: Left: Examples and GT-masks from the proposed *an8FLower* dataset. Center: Comparison of generated explanations. Right: Mean intersection over union (mIoU) of the evaluated methods.

on prediction confidence of 2% w.r.t. to [24]. This suggest that our method is able to maintain focus on relevant class features while producing detailed heatmaps with better visual quality.

4.3 Measuring Visual Explanation Accuracy

We generate two synthetic datasets, *an8Flower-single-6c* and *an8Flower-double-12c*, with 6 and 12 classes respectively. In the former, a fixed single part of the object is allowed to change color. This color defines the classes of interest. In the latter, a combination of color and the part on which it is located defines the discriminative feature. For this second case, an explanation should focus on both color and shape features (parts). After defining these features, we generate masks that overlap with the discriminative regions. Then, we threshold the heatmaps at given values and measure the pixel-level intersection over union (IoU) of a model explanation (produced by the method to be evaluated) w.r.t. these masks. We test a similar model as for the MNISTdataset (Sec. 4.1) trained on each variant of the *an8Flower* dataset. In Fig. 9 we report the performance of the proposed feature selection method using three different means (Upsamp. Act. Maps, Deconv+GB [24] and ours heatmap variant) to generate heatmap visualizations. See Fig. 9 (left) for examples.

We can notice in Fig. 9 (center) that our explanations effectively highlight the pre-defined discriminative regions regardless of whether they are related to color and/or shape. Moreover, our visualizations have a better coverage than those produced by upsampled activation maps. These observations are further confirmed by the quantitative evaluation (Fig. 8 & Fig. 9 (right)). The quantitative results show that our method has higher mean IoU coverage of the discriminative features when compared to existing methods. Please refer to the supplementary material for more details and visualizations.

5 Conclusion

We propose a method to enrich the prediction made by deep neural networks by indicating the features that contributed to such prediction. This enables our method with visual *explanation* capabilities. Our method identifies features internally encoded by the network that are relevant for the task originally addressed by the network. Moreover, it allows *interpretation* of these features by the generation of average feature-wise visualizations. In addition, we have proposed a method to attenuate the artifacts introduced by strided operations in visualizations made by Deconvnet-based methods. This empowers our method with richer visual feedback with pixel-level precision without requiring additional annotations for supervision. Finally, we have proposed a novel dataset specifically designed for the evaluation of methods for explanation of deep neural networks.

References

[1] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7):e0130140, 07 2015.

[2] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[3] Bert De Brabandere, Xu Jia, Tinne Tuytelaars, and Luc Van Gool. Dynamic filter networks. In *Neural Information Processing Systems (NIPS)*, 2016.

[4] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: delving deep into convolutional nets. In *British Machine Vision Conference (BMVC)*, 2014.

[5] Carl Doersch, Saurabh Singh, Abhinav Gupta, Josef Sivic, and Alexei A Efros. What makes paris look like paris? *Communications of the ACM*, 58(12):103–110, 2015.

[6] V. Escorcia, J. C. Niebles, and B. Ghanem. On the relationship between visual attributes and convolutional networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[7] R. Fong and A. Vedaldi. Net2Vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[8] Arnab Ghosh, Viveka Kulharia, Vinay P. Namboodiri, Philip H. S. Torr, and Puneet Kumar Dokania. Multi-agent diverse generative adversarial networks. In *arXiv 1704.02906*, 2017.

[9] Abel Gonzalez-Garcia, Davide Modolo, and Vittorio Ferrari. Do semantic parts emerge in convolutional neural networks? *International Journal of Computer Vision (IJCV)*, pages 1–19, 2017.

[10] F. Grün, C. Rupprecht, N. Navab, and F. Tombari. A taxonomy and library for visualizing learned features in convolutional neural networks. In *International Conference on Machine Learning (ICML) Workshops*, 2016.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[12] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating visual explanations. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *European Conference on Computer Vision (ECCV)*, 2016.

[13] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):664–676, April 2017.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Neural Information Processing Systems (NIPS)*, 2012.

[15] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[16] Yao Li, Lingqiao Liu, Chunhua Shen, and Anton van den Hengel. Mining mid-level visual patterns with deep cnn activations. *International Journal of Computer Vision (IJCV)*, 121(3):344–364, 2017.

[17] Julien Mairal, Francis Bach, and Jean Ponce. Sparse modeling for image and vision processing. *Found. Trends. Comput. Graph. Vis.*, 8(2-3):85–283, 2014.

[18] José Oramas M. and Tinne Tuytelaars. Modeling visual compatibility through hierarchical mid-level elements. *CoRR*, abs/1604.00036, 2016.

[19] Konstantinos Rematas, Basura Fernando, Frank Dellaert, and Tinne Tuytelaars. Dataset fingerprints: Exploring image collections through data mining. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[20] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *Neural Information Processing Systems (NIPS)*, 2015.

[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[22] E. Simo-Serra, S. Fidler, F. Moreno-Noguer, and R. Urtasun. Neuroaesthetics in fashion: modeling the perception of fashionability. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[23] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *International Conference on Learning Representations (ICLR) Workshops*, 2014.

[24] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller. Striving for simplicity: the all convolutional net. In *International Conference on Learning Representations (ICLR) Workshops*, 2015.

[25] E. van den Berg and M. P. Friedlander. Probing the pareto frontier for basis pursuit solutions. *SIAM J. Sci. Comput.*, 31(2):890–912, 2008.

[26] Kaili Wang, Yu-Hui Huang, José Oramas M, Luc Van Gool, and Tinne Tuytelaars. An analysis of human-centered geolocation. *Winter Conference on Applications of Computer Vision (WACV)*, 2018.

[27] Ning Xie, Md. Kamruzzaman Sarker, Derek Doran, Pascal Hitzler, and Michael Raymer. Relating input concepts to convolutional neural network decisions. In *Neural Information Processing Systems (NIPS) Workshops*, 2017.

[28] Jason Yosinski, Jeff Clune, Anh Mai Nguyen, Thomas J. Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. 2015.

[29] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, 2014.

[30] X. Zeng, W. Ouyang, J. Yan, H. Li, T. Xiao, K. Wang, Y. Liu, Y. Zhou, B. Yang, Z. Wang, H. Zhou, and X. Wang. Crafting gbd-net for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(99):1–1, 2017.

[31] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable convolutional neural networks. *CoRR*, abs/1710.00935, 2017.

[32] B. Zhou, A. Khosla, Lapedriza. A., A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[33] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. In *International Conference on Learning Representations (ICLR)*, 2015.