CNN Fixations: An unraveling approach to visualize the discriminative image regions

Konda Reddy Mopuri*, Utsav Garg*, R. Venkatesh Babu, *Senior Member, IEEE*

***Abstract*—Deep convolutional neural networks (CNN) have revolutionized the computer vision research and have seen unprecedented adoption for multiple tasks such as classification, detection, caption generation, etc. However, they offer little transparency into their inner workings and are often treated as black boxes that deliver excellent performance. In this work, we aim at alleviating this opaqueness of CNNs by providing visual explanations for the network's predictions. Our approach can analyze a variety of CNN based models trained for computer vision applications such as object recognition and caption generation. Unlike existing methods, we achieve this via unraveling the forward pass operation. The proposed method exploits feature dependencies across the layer hierarchy and uncovers the discriminative image locations that guide the network's predictions. We name these locations CNN-Fixations, loosely analogous to human eye fixations. Our approach is a generic method that requires no architectural changes, additional training or gradient computation and computes the important image locations (CNN Fixations). We demonstrate through a variety of applications that our approach is able to localize the discriminative image locations across different network architectures, diverse vision tasks and data modalities.**

***Index Terms*—Explainable AI, CNN visualization, visual explanations, label localization, weakly supervised localization**

I. Introduction

CONvolutional Neural Networks (CNN) have demonstrated outstanding performance for a multitude of computer vision tasks ranging from recognition and detection to image captioning. CNNs are complex models to design and train. They are non-linear systems that almost always have numerous local minima and are often sensitive to the training parameter settings and initial state. With time, these networks have evolved to have better architectures along with improved regularizers to train them. For example, in case of recognition, from AlexNet [15] in 2012 with 8 layers and $60M$ parameters, they advanced to ResNets [10] in 2015 with hundreds of layers and $1.7M$ parameters. Though this has resulted in a monotonic increase in performance on many vision tasks (e.g. recognition on ILSVRC [23], semantic segmentation on PASCAL [8]), the model complexity has increased as well.

In spite of such impressive performance, CNNs continue to be complex machine learning models which offer limited transparency. Current models shed little light on explaining why and how they achieve higher performance and as a result are treated as black boxes. Therefore, it is important to understand what these networks learn in order to gain insights into their representations. One way to understand CNNs is to look at the important image regions that influence their predictions. In cases where the predictions are inaccurate, they should be able to offer visual explanations (as shown in Fig.8) in terms of the regions responsible for misguiding the CNN. Visualization can play an essential role in understanding CNNs and in devising new design principles (e.g., architecture selection shown in [35]). With the availability of rich tools for visual exploration of architectures during training and testing, one can reduce the gap between theory and practice by verifying the expected behaviours and exposing the unexpected behaviours that can lead to new insights. Towards this, many recent works (e.g. [25], [27], [29], [36], [37]) have been proposed to visualize CNNs' predictions. The common goal of these works is to supplement the label predicted by the classifier (CNN) with the discriminative image regions (as shown in Fig. 4 and Fig. 6). These maps act as visual explanations for the predicted label and make us understand the class specific patterns learned by the models. Most of these methods utilize the gradient information to visualize the discriminative regions in the input that led to the predicted inference. Some (e.g. [37]) are restricted to work for specific network architectures and output low resolution visualization maps that are interpolated to the original input size.

On the other hand, we propose a visualization approach that exploits the learned feature dependencies between consecutive

Fig. 1: CNN fixations computed for a pair of sample images from ILSVRC validation set. Left column: input images. Middle column: corresponding CNN fixations (locations shown in red) overlaid on the image. Right column: The localization map computed form the CNN fixations via Gaussian blurring.

* denotes equal contribution.

Konda Reddy Mopuri (kondamopuri@iisc.ac.in) and R. Venkatesh Babu (venky@iisc.ac.in) are with Video Analytics Lab, Department of Computational and Data Sciences, Indian Institute of Science, Bengaluru, India, 560012. Utsav Garg (utsav002@e.ntu.edu.sg) was an intern at Video Analytics Lab, CDS, IISc, Bangaluru from NTU, Singapore.

layers of a CNN using the forward pass operations. That is, in a given layer, for a chosen neuron activation, we can determine the set of positively correlated activations from the previous layer that act as evidence. We perform this process iteratively from the softmax layer till the input layer to determine the discriminative image pixels that support the predicted inference (label). In other words, our approach locates the image regions that were responsible for the CNN's prediction. We name them *CNN fixations*, loosely analogous to the human eye fixations. By giving away these regions, our method makes the CNNs more expressive and transparent by offering the needed visual explanations. We highlight (as shown in Fig. 1) the discriminative regions by tracing back the corresponding label activation via strong neuron activation paths onto the image plane. Note that we can visualize not only the label activations present in the softmax layer but also any neuron in the model's architecture. Our method offers a high resolution, pixel level localizations. Despite the simplicity of our approach, it could reliably localize objects in case networks trained for recognition task across different input modalities (such as images and sketches) and uncover objects responsible for the predicted caption in case of caption generators (e.g. [32]).

The major contributions of this paper can be listed as follows:

- A simple yet powerful method that exploits feature dependencies between a pair of consecutive layers in a CNN to obtain discriminative pixel locations that guide its prediction.
- We demonstrate using the proposed approach that CNNs trained for various vision tasks (e.g. recognition, captioning) can reliably localize the objects with little additional computations compared to the gradient based methods.
- We show that the approach generalizes across different generations of network architectures and across different data modalities. Furthermore, we demonstrate the effectiveness of our method through a multitude of applications.

Rest of this paper is organized as follows: section II presents and discusses existing works that are relevant to the proposed method, section III presents the proposed approach in detail, section IV demonstrates the effectiveness of our approach empirically on multiple tasks, modalities and deep architectures, and finally section V presents the conclusions.

II. Related Work

Our approach draws similarities to recent visualization works. A number of attempts (e.g. [3], [22], [25], [27], [29], [35]–[37]) have been made in recent time to visualize the classifier decisions and deep learned features.

Most of these works are gradient based approaches that find out the image regions which can improve the predicted score for a chosen category. Simonyan *et al.* [27] measure sensitivity of the classification score for a given class with respect to a small change in pixel values. They compute partial derivative of the score in the pixel space and visualize them as saliency maps. They also show that this is closely related to visualizing using deconvolutions by Zeiler *et al* [35]. The deconvolution [35] approach visualizes the features (visual concepts) learned by the neurons across different layers. Guided backprop [29] approach modifies the gradients to improve the visualizations qualitatively.

Zhou *et al.* [37] showed that class specific activation maps can be obtained by combining the feature maps before the GAP (Global Average Pooling) layer according to the weights connecting the GAP layer to the class activation in the classification layer. However, their method is architecture specific, restricted to networks with GAP layer. Selvaraju *et al.* [25] address this issue by making it a more generic approach utilizing gradient information. Despite this, [25] still computes low resolution maps (e.g. 13×13). Majority of these methods compute partial derivatives of the class scores with respect to image pixels or intermediate feature maps for localizing the image regions.

Another set of works (e.g. [3], [6], [22], [38]) take a different approach and assign a relevance score for each feature with respect to a class. The underlying idea is to estimate how the prediction changes if a feature is absent. Large difference in prediction indicates that the feature is important for prediction and small changes do not affect the decision. In [6], authors find out the probabilistic contribution of each image patch to the confidence of a classifier and then they incorporate the neighborhood information to improve their weakly supervised saliency prediction. Zhang *et al.* [36] compute top down attention maps at different layers in the neural networks via a probabilistic winner takes all framework. They compute marginal winning probabilities for neurons at each layer by exploring feature expectancies. At each layer, the attention map is computed as the sum of these probabilities across the feature maps.

Unlike these existing works, the proposed approach finds the responsible pixel locations by simply unraveling the underlying forward pass operations. Starting from the neuron of interest (e.g. the predicted category label), we rely only on the basic convolution operation to figure out the visual evidence offered by the CNNs. Most of the existing works (e.g. [29], [35]) realize the discriminative regions via reconstructing the chosen activation. Whereas, our method obtains a binary output at every layer via identifying the relevant neurons. At each layer we can obtain a heat map by simple Gaussian blurring of the binary output. Note that the proposed *CNN-fixations* method has no hyper-parameters or heuristics in the entire process of back tracking the evidence from the softmax layer onto the input image. Fundamentally, our approach exploits the excitatory nature of neurons, which is, being positively correlated and to fire for a specific stimulus (input) from the preceding layer. Though the proposed approach is simple and intuitive in nature, it yields accurate and high resolution visualizations. Unlike majority of the existing works, the proposed method does not require to perform gradient computations, prediction differences, winning probabilities for neurons. Also, the proposed approach poses no architectural constraints and just requires a single forward pass and backtracking operations for the selected neurons that act as the evidence.

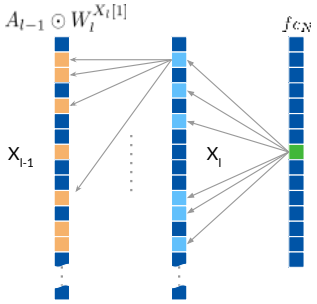


Fig. 2: Evidence localization shown between a pair of fc layers. Note that in layer $l-1$, operation is shown for one discriminative location in X_l. The dark blue color in layers $l-1$ and l indicates locations with $C < 0$.

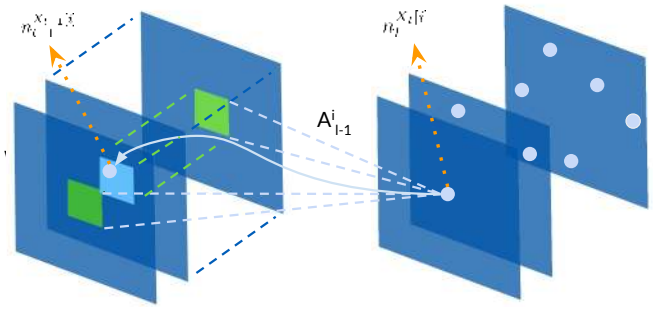


Fig. 3: Evidence localization between a pair of convolution layers l and $l-1$. $A_{l-1}^{X_l[i]}$ is the receptive field corresponding to $X_l[i]$. Note that $W_l^{X_l[i]}$ is not shown, however the channel (feature) with maximum contribution (shown in light blue) is determined based on $A_{l-1}^{X_l[i]} \odot W_l^{X_l[i]}$.

III. Proposed Approach

In this section, we describe the underlying operations in the proposed approach to determine the discriminative image locations that guide the CNN to its prediction. Note that the objective is to provide visual explanations for the predictions (e.g. labels or captions) in terms of the important pixels in the input image (as shown in Fig. 1 and 6).

Typical deep network architectures have basic building blocks in the form of fully connected, convolution, skip connections and pooling layers or LSTM units in case of captioning networks. In this section we explain our approach for tracing the visual evidence for the prediction across these building blocks onto the image. The following notation is used to explain our approach: we start with a neural network with N layers, thus, the layer indices l range from $1, 2, \ldots N$. At layer l, we denote the activations as A_l and weights connecting this layer from previous layer as W_l. Also, n_l^k represents k^{th} neuron at layer l. X_l is the vector of discriminative locations in the feature maps at layer l and m is its cardinality. Note that the proposed approach is typically performed during inference (testing) to provide the visual explanations for the network's prediction.

A. Fully Connected

A typical CNN for recognition contains a fully connected (fc) layer as the final layer with as many neurons as the number of categories to recognize. During inference, after a forward pass of an image through the CNN, we start with X_N being a vector with one element in the final fc layer, which is the predicted label (shown as green activation in Fig. 2). Note that this can be any chosen activation (neuron) in any layer of the network, our visualization method imposes no restrictions and can localize all the neurons in the architecture.

In case of stacked fc layers, the set X_{l-1} for an fc layer $(l-1)$ will be a vector of indices belonging to important neurons $[n_{l-1}^{X_{l-1}[1]} \ldots n_{l-1}^{X_{l-1}[m]}]$ chosen by the succeeding (higher) layer l. This set is the list of all neurons in A_{l-1} that contribute to the elements in X_l (in higher layer). That is, for each of the important (discriminative) features determined in the higher layer (X_l), we find the evidence in the current layer (A_{l-1}). Thus the proposed approach finds out the evidence by exploiting the feature dependencies between layer l and $l-1$ learned during the training process. We consider all the neurons in A_{l-1} that aid positively (excite) for the task in layer l as its evidence. Algorithm 1 explains the process of tracing the evidence from a fully connected layer onto the preceding layer.

In case the layer is preceded by a spatial layer (convolution or pooling), we flatten the $3D$ activations A_{l-1} to get a vector for finding the discriminative neurons, finally we convert the indices back to $3D$. Therefore, for spatial layers, X_{l-1} is a list with each entry being three dimensional, namely, $\{ feature\ (or\ channel),\ x,$ and $y\}$. Figure 2 shows how we determine the evidence in the preceding layer for important neurons of an fc layer.

Typically during the evidence tracing, after reaching the first fc layer, a series of conv/pool layers will be encountered. The next subsection describes the process of evidence tracing through a series of convolution layers.

Algorithm 1: Discriminative Localization at fc layers.

input: X_l, incoming discriminative locations from higher layer : $\{ X_l[1], \ldots, X_l[m] \}$
W_l, weights of higher layer l
A_{l-1}, activations at current layer $l-1$
output: X_{l-1}, outgoing discriminative locations from the current layer

```
X_{l-1} = φ
for i=1:m do
    W_l^{X_l[i]} ← weights of neuron n_l^{X_l[i]}
    C ← A_{l-1} ⊙ W_l^{X_l[i]}    // Hadamard product
    X_{l-1} ← append ( X_{l-1}, args(C > 0) )
end
```

B. Convolution

As discussed in the previous subsection, upon reaching a spatial layer, X_l will be a set of $3D$ indices specifying the location of each discriminative neuron. This subsection explains

how the proposed approach handles the backtracking between spatial layers. Note that a typical pooling layer will have a $2D$ receptive field and a convolution layer will have a $3D$ receptive field to operate on the previous layer's output. For each important location in X_l, we extract the corresponding receptive field activation $A_{l-1}^{X_l[i]}$ in layer $l-1$ (shown as green cuboid in Fig. 3). Hadamard product $(A_{l-1}^{X_l[i]} \odot W_l^{X_l[i]})$ is computed between this receptive activations and the filter weights of the neuron $n_l^{X_l[i]}$. We then find out the feature (channel) in A_{l-1} that contributes highest (shown in light blue color in Fig. 3) by adding the individual activations from that channel in the hadamard product. That is because, the sum of these terms in the hadamard product gives the contribution of the corresponding feature to excite the discriminative activation in the succeeding layer.

Algorithm 2 explains this process for convolution layers. In the algorithm, k_{l-1} denotes the kernel size of the convolution filters, ${A_{l-1}}^{X_l[i]}$ are the receptive activations in the previous layer, and hence is a $3D$ spatial blob. Therefore, when the Hadamard product is computed with the weights $(W_l^{X_l[i]})$ of the neuron, the result is also a spatial blob of the same size. We sum the output across x and y directions to locate the most discriminative feature map "ch" (shown in light blue color in Fig. 3). During this transition, spatial location of the activation can also get affected. That means, (x, y) location in the succeeding layer is traced onto the strongest contributing activation of channel "ch" in the current layer. Instead, we can also trace back to the same location within the most contributing channel "ch". However, we empirically found that this is not significantly different from the former alternative. Therefore, in all our experiments, for computational efficiency, we follow the latter alternative of tracking onto the same location as in the succeeding layer. Note that the procedures we follow for evidence tracking across fc (Algo. 1) and $conv$ (Algo. 2) layers are fundamentally similar, except that $conv$ layers operate over $3D$ input blobs, whereas fc layers have a $1D$ input blob (after vectorizing). Algorithm 2 explains the process considering the localized input blobs (receptive activations) and the convolution kernels to the exact implementation details.

In case of pooling layers, we extract the $2D$ receptive neurons in the previous layer and find the location with the highest activation. This is because most of the architectures typically use max-pooling to sub-sample the feature maps. The activation in the succeeding layer is the maximum activation present in the corresponding receptive field in the current layer. Thus, when we backtrack an activation across a sub-sampling layer, we locate the maximum activation in its receptive field.

Thus for a CNN trained for recognition, the proposed approach starts from the predicted label in the final layer and iteratively backtracks through the fc layers and then through the convolution layers onto the image. CNN Fixations (red dots shown in middle column of Fig. 1) are the final discriminative locations determined on the image. Note that the fixations are $3D$ coordinates since the input image generally contains three channels (R, G and B). However, we consider the union of spatial coordinates (x and y) of the fixations neglecting the channel.

Algorithm 2: Discriminative Localization at Convolution layers

input: X_l, incoming discriminative locations from higher layer : $X_l[1] \dots X_l[m]$
W_l, weights at layer l
A_{l-1}, activations at layer $l-1$
output: X_{l-1}, outgoing discriminative locations in the current layer

$S(.)$: a function that sums a tensor along xy axes
$X_{l-1} = \phi$
for *i=1:m* **do**
  $W_l^{X_l[i]} \leftarrow$ weights for neuron $n_l^{X_l[i]}$
  $A_{l-1}^{X_l[i]} \leftarrow$ receptive activations for neuron $n_l^{X_l[i]}$
  $C \leftarrow S(A_{l-1}^{X_l[i]} \odot W_l^{X_l[i]})$ // Per channel contributions
  $ch \leftarrow argmax(C)$// Discriminative channel
  $(P_x, P_y) \leftarrow argmax(A_{l-1}^{X_l[i]}(:,:,ch)) \odot W_l^{X_l[i]}(:,:,ch))$ // Discriminative location in channel 'ch'
  $X_{l-1} \leftarrow$ append $(X_{l-1}, ch.k_{l-1}^2 + P_x.k_{l-1} + P_y)$
end
$X_{l-1} \leftarrow unique(X_{l-1})$

C. Advanced architectures: Inception, Skip connections and Densely connected convolutions

Inception modules have shown (e.g. Szegedy *et al.* [30]) to learn better representations by extracting multi-level features from the input. They typically comprise of multiple branches which extract features at different scales and concatenate them along the channels at the end. The concatenation of feature maps will have a single spatial resolution but increased depth through multiple scales. Therefore, each channel in 'Concat' is contributed by exactly one of these branches. Let us consider an inception layer with activation A_l with B input branches getting concatenated. That means, A_l is concatenation of B outputs obtained via convolving the previous activations A_{l-1} with a set of B different weights $\{W_{lb}\}$ where $b \in \{1, \dots B\}$. For each of the important activations X_l in the inception layer, there is exactly one input branch connecting it to the previous activations A_{l-1}. Since we know the number of channels resulted by each of the input branches, we can identify the corresponding input branch for X_l from the channel on which it lies. Once we determine the corresponding input branch, it is equivalent to performing evidence tracing via a $conv$ layer. Hence, we perform the same operations as we perform for a $conv$ layer (discussed in section III-B and Algorithm 2) after determining which branch caused the given activation.

He *et al.* [10] presented a residual learning framework to train very deep neural networks. They introduced the concept of residual blocks (or ResBlocks) to learn residual function with respect to the input. A typical ResBlock contains a skip path and a residual (delta) path. The delta path (D_{l-1}) generally consists of multiple convolutional layers and the skip path is an identity connection with no transformation. Ending of the ResBlock performs element wise sum of the incoming

skip (A_{l-1}) and delta branches (D_{l-1}). Note that this is unlike the inception block where each activation is a contribution of a single transformation. Therefore, for each discriminative location in X_l, we find the branch (either skip or delta) that has a higher contributing activation and trace the evidence through that route. If for a given location $X_l[i]$, the skip path contributes more to the summation, it is traced directly onto A_{l-1} through the identity transformation. On the other hand, if the delta path contributes more than the skip connection, we trace through the multiple $conv$ layers of the delta path as we explained in section III-B. We perform this process iteratively across all the ResBlocks in the architecture to determine the visual explanations.

Huang *et al.* introduced Dense Convolutional Networks (DenseNet [12]) that connects each layer to every other layer in a feed-forward fashion. For each layer, feature maps of all the earlier layers are used as input and its own feature maps are used as input to later layers. In other words, dense connections can be considered as a combination of skip connections and inception modules. At a given layer (l) in the architecture, a skip path from the previous layer's output (A_{l-1}) gets concatenated to the activations (feature maps) computed at this layer (A_l). Note that in case of ResNets, the skip and and delta paths gets added. Therefore, for a given discriminative activation in the current layer, the backtracking has two options: either it belongs to the current feature maps computed at this layer or it is transferred from the previous layer. If it belongs to current set of feature maps, we can backtrack using the conv component (section. III-B) of the proposed approach. Else, if it belongs to the feature maps copied from the previous layers, we simply transfer (copy) the discriminative location onto the previous layer, since it is an identity transformation from previous layer to current layer. This process of evidence tracing is performed iteratively till the input layer to obtain the CNN-Fixations. Thus, our method is a generic approach and it can visualize all CNN architectures ranging from the first generation AlexNet [15] to the recent DenseNets [12].

D. LSTM Units

In this subsection we discuss our approach to backtrack through an LSTM [11] unit used in caption generation networks (e.g. [32]). The initial input to the LSTM unit is random state and the image embedding I encoded by a CNN. In the following time steps image embedding is replaced by embedding for the word predicted in the previous time step. An LSTM unit is guided by the following equations [32]:

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1}) \tag{1}$$

$$f_t = \sigma(W_{fx}x_t + W_{fm}m_{t-1}) \tag{2}$$

$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1}) \tag{3}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot h(W_{cx}x_t + W_{cm}m_{t-1}) \tag{4}$$

$$m_t = o_t \odot c_t \tag{5}$$

Here, i, f and o are the input, forget and output gates respectively of the LSTM and σ and h are the sigmoid and hyperbolic-tan non-linearities. m_t is the state of the LSTM which is passed along with the input to the next time step. At each time step, a softmax layer is learned over m_t to output a probability density over a set of dictionary words.

Our approach takes the maximum element in m at the last unrolling and then tracks back the discriminative locations through the four gates individually and then accumulates them as locations on m_{t-1}. Tracking back through these gates involves operations similar to the ones discussed in case of fully connected layers III-A. We iteratively perform backtracking through the time steps till we finally reach the image embedding I. Once we reach I, we perform the operations discussed in sections III-A and III-B to obtain the discriminative locations on the image.

IV. Applications

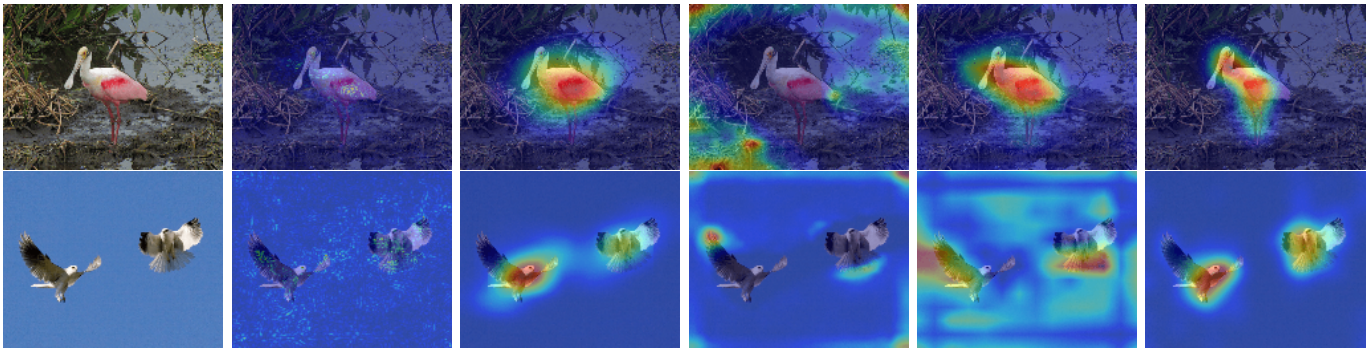
This section demonstrates the effectiveness of the proposed approach across multiple vision tasks and modalities through a variety of applications.

The proposed approach is both network and framework agnostic. It requires no training or modification to the network to get the discriminative locations. The algorithm needs to extract the weights and activations from the network to perform the operations discussed in the sections above. Therefore any network can be visualized with any deep learning framework. For the majority of our experiments we used the Python binding of Caffe [13] to access the weights and activations, and we used Tensorflow [1] in case of captioning networks as the models for Show and Tell [32] are provided in that framework. After finding the important pixels in the image, we perform outlier removal before we compute the heat map. We consider a location to be an outlier, if it is not supported by sufficient neighboring fixations. Specifically, if a fixation has less than certain percentage of total fixations within a given circle around it, we neglect it. Codes for the project are publicly available at https://github.com/val-iisc/cnn-fixations. Additional qualitative results for some applications are available at http://val.cds.iisc.ac.in/cnn-fixations/.

A. Weakly Supervised Object Localization

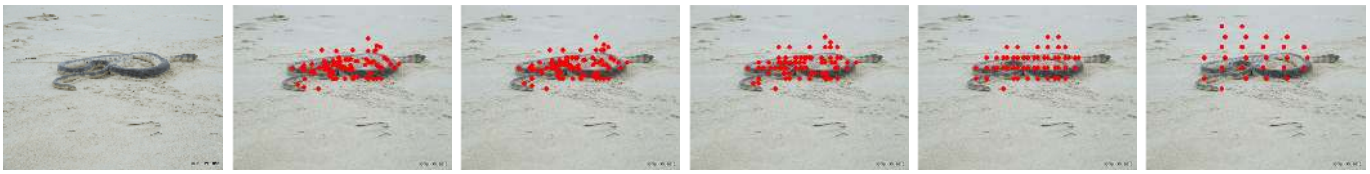
We now empirically demonstrate that the proposed CNN fixations approach is capable of efficiently localizing the object recognized by the CNN. Object recognition or classification is the task of predicting an object label for a given image. However, object detection involves not only predicting the object label but also localizing it in the given image with a bounding box. The conventional approach has been to train CNN models separately for the two tasks. Although some works (e.g. [21], [33]) share features between both tasks, detection models (e.g. [21], [26], [33]) typically require training data with human annotations of the object bounding boxes. In this section, we demonstrate that the CNN models trained to perform object recognition are also capable of localization.

For our approach, after the forward pass, we backtrack the label on to the image. Unlike other methods our approach finds the important locations (as shown in Figure 1) instead of a heatmap, therefore we perform outlier removal as follows:



Input Image | Backprop [27] | CAM [37] | cMWP [36] | Grad-CAM [25] | Proposed

Fig. 4: Comparison of the localization maps for sample images from ILSVRC validation set across different methods for VGG-16 [28] architecture without any thresholding of maps. For our method, we blur the discriminative image locations using a Gaussian to get the map.



Input Image | Pool1 | Pool2 | Pool3 | Pool4 | Pool5

Fig. 5: CNN-Fixations at intermediate layers of VGG-16 [28] network. Note that the fixations at deeper layers are also displayed on the original resolution image via interpolating.

TABLE I: Error rates for Weakly Supervised Localization of different visualization approaches on ILSVRC validation set. The numbers show error rate, which is $100 -$ Accuracy of localization (lower the better). (*) denotes modified architecture, bold face is the best performance in the column and underline denotes the second best performance in the column. Note that the numbers are computed for the top-1 recognition prediction.

Method	AlexNet	VGG-16	GoogLeNet	ResNet-101	DenseNet-121
Backprop	**65.17**	61.12	61.31	57.97	67.49
CAM	67.19*	57.20*	60.09	**48.34**	**55.37**
cMWP	72.31	64.18	69.25	65.94	64.97
Grad-CAM	71.16	56.51	74.26	64.84	75.29
Ours	65.70	**55.22**	**57.53**	54.31	56.72

we consider a location to be an outlier if the location is not sufficiently supported by neighboring fixation locations. Particularly, if any of the *CNN Fixations* has less than a certain percentage of the fixations present in a given circle around it, we consider it as an outlier and remove it. These two parameters, percentage of points and radius of the circle were found over a held out set, and we found 5% of points and radius equal to $9-11\%$ of the image diagonal to perform well depending on the architecture. After removing the outliers, we use the best fitting bounding box for the remaining locations as the predicted location for the object.

We perform localization experiments on the ILSVRC [23] validation set of $50,000$ images with each image having one or multiple objects of a single class. Ground truth consists of object category and the bounding box coordinates for each instance of the objects. Similar to the existing visualization methods (e.g. [25], [36], [37]), our evaluation metric is accuracy of localization, which requires to get the prediction correct and obtain a minimum of 0.5 Intersection over Union (IoU) between the predicted and ground-truth bounding boxes. Table I shows the error rates $(100 -$ Accuracy of localization$)$ for corresponding visualization methods across multiple network architectures such as AlexNet [15], VGG [28], GoogLeNet [30], ResNet [10] and DenseNet [12]. Note that the error rates are computed for the $top-1$ recognition prediction. In order to obtain a bounding box from a map, each approach uses a different threshold. For CAM [37] and Grad-CAM [25] we used the threshold provided in the respective papers, for Backprop (for ResNet and DenseNet, other values from CAM) and cMWP [36] we found the best performing thresholds on the same held out set. The values marked with $*$ for CAM are for a modified architecture where all fc layers were replaced with a GAP layer and the model was retrained with the full ILSVRC training set ($1.2M$ images). Therefore, these numbers are not comparable. This is a limitation for CAM as it works only for networks with GAP layer and in modifying the architecture as explained above it loses recognition performance by 8.5% and 2.2% for AlexNet and VGG respectively.

Figure 4 shows the comparison of maps between different approaches. The Table I shows that the proposed approach performs consistently well across a contrasting range of architectures, unlike other methods which perform well on selected architectures. Also, in Figure 5 we present visualization at various layers in the architecture of VGG-16 [28]. Specifically,

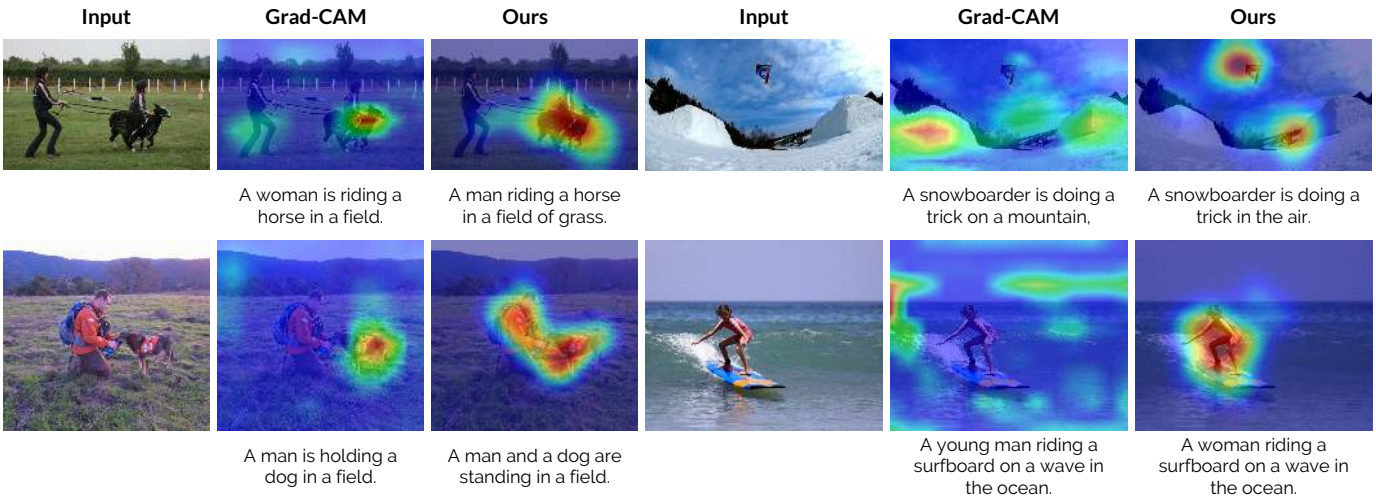


Fig. 6: Discriminative localization obtained by the proposed approach for captions predicted by the Show and Tell [32] model on sample images from MS COCO [16] dataset. Grad-CAM's illustrations are for Neuraltalk [14] model. Note that the objects predicted in the captions are better highlighted for our method.

we show the evidence at all the five pool layers. Note that the fixations are computed on the feature maps which are of lower resolution compared to the input image. Therefore it is required to interpolate the location of fixations in order to show them on the input image. Observe that the localization improves as the resolution of the feature map increases, i.e., towards the input layer, fixations become more dense and accurate.

B. Grounding Captions

In this subsection, we show that our method can provide visual explanations for image captioning models. Caption generators predict a human readable sentence that describes contents of a given image. We present qualitative results for getting localization for the whole caption predicted by the Show and Tell [32] architecture.

The architecture has a CNN followed by an LSTM unit, which sequentially generates the caption word by word. The LSTM portion of the network is backtracked as discussed in section III-D following which we backtrack the CNN as discussed in sections III-A and III-B.

Figure 6 shows the results where all the important objects that were predicted in the caption have been localized on the image. This shows that the proposed approach can effectively localize discriminative locations even for caption generators (i.e, grounding the caption). Our approach generalizes to deep neural networks trained for tasks other than object recognition. Note that some of the existing approaches discussed in the previous sections do not support localization for captions in their current version. For example, CAM [37] requires a GAP layer in the model and Ex-BP [36] grounds the tags predicted by a classification model instead of working with a caption generator.

C. Saliency

We now demonstrate the effectiveness of the proposed approach for predicting weakly-supervised saliency. The objective of this task is similar to that of Cholakkal *et al.* [6], where we perform weakly supervised saliency prediction using the models trained for object recognition. The ability of the proposed approach to provide visual explanations via back tracking the evidence onto the image is exploited for salient object detection.

Following [6], we perform the experiments on the Graz-2 [17] dataset. Graz-2 dataset consists of three classes namely bike, car and person. Each class has 150 images for training and same number for testing. We fine-tuned VGG-16 architecture for recognizing these 3 classes by replacing the final layer with 3 units. We evaluated all approaches discussed in section IV-A in addition to [6]. In order to obtain the saliency map from the fixations, we perform simple Gaussian blurring on the obtained CNN fixations. All the maps were thresholded based on the best thresholds we found on the train set for each approach. The evaluation is based on pixel-wise precision at equal error rate (EER) with the ground truth maps.

Table II presents the precision rates per class for the Graz-2 dataset. Note that CAM [37] was excluded as it does not work with the vanilla VGG [28] network. This application highlights that the approaches which obtain maps at a low resolution and up-sample them to image size perform badly in this case due to the pixel level evaluation. However, our approach outperforms other methods to localize salient image regions by a huge margin.

TABLE II: Performance of different visualization methods for predicting saliency on Graz-2 dataset. Numbers denote the Pixel-wise precision at EER.

Method	Bike	Car	Person	Mean
Backprop	39.51	28.50	42.64	36.88
cMWP	61.84	46.82	44.02	50.89
Grad-CAM	65.70	6.58	7.98	60.09
WS-SC [6]	7.5	56.48	57.56	0.52
Ours	**71.21**	**62.15**	**61.27**	**64.88**

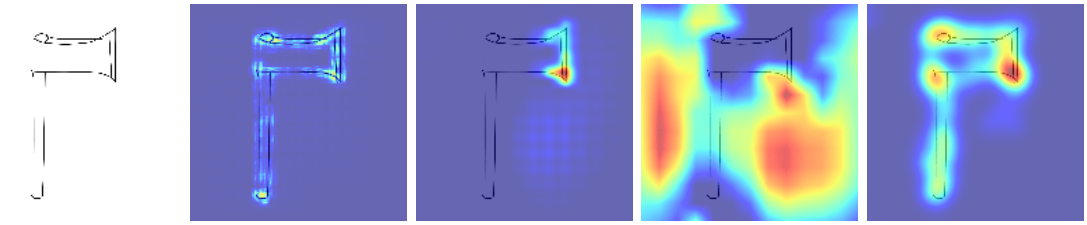


Fig. 7: Comparison of localization maps with different methods for a sketch classifier [24].

D. Localization across modalities

We demonstrate that the proposed approach visualizes classifiers learned on other modalities as well. We perform the proposed CNN Fixations approach to show visualizations for a sketch classifier from [24]. Sketches are a very different data modality compared to images. They are very sparse depictions of objects with only edges. CNNs trained to perform recognition on images are fine-tuned [24], [34] to perform recognition on sketches. We have considered AlexNet [15] fine-tuned over 160 categories of sketches from Eitz dataset [7] to visualize the predictions.

Figure 7 shows the localization maps for different approaches. We can clearly observe that the proposed approach highlights all the edges present in the sketches. This shows that our approach effectively localizes the sketches much better than the compared methods, showing it generalizes across different data modalities.

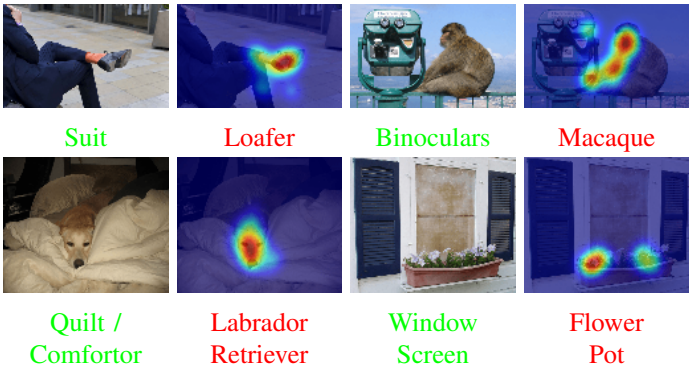


Fig. 8: Explaining the wrong recognition results obtained by VGG [28]. Each pair of images along the rows show the image and its corresponding fixation map. Ground truth label is shown in green and the predicted is shown in red. Fixations clearly provide the explanations corresponding to the predicted labels.

E. Explanations for erroneous predictions by CNNs

CNNs are complex machine learning models offering very little transparency to analyse their inferences. For example, in cases where they wrongly predict the object category, it is required to diagnose them in order to understand what went wrong. If they can offer a proper explanation for their predictions, it is possible to improve various aspects of training and performance. The proposed CNN-fixations can act as a tool to help analyse the training process of CNN models. We demonstrate this by analysing the misclassified instances for object recognition. In Figure 8 we show sample images from ILSVRC validation images that are wrongly classified by VGG [28]. Each image is associated with the density map computed by our approach. Below each image-and-map pair, the ground truth and predicted labels are displayed in green and red respectively. Multiple objects are present in each of these images. The CNN recognizes the objects that are not labeled but are present in the images. The computed maps for the predicted labels accurately locate those objects such as loafer, macaque, etc. and offer visual explanations for the CNNs behaviour. It is evident that these images are labeled ambiguously and the proposed method can help improve the annotation quality of the data.

F. Presence of Adversarial noise

Many recent works (e.g. [9], [18], [19]) have demonstrated the susceptibility of convolutional neural networks to *Adversarial* samples. These are images that have been perturbed with structured quasi-imperceptible noise towards the objective of fooling the classifier. Figure 9 shows two samples of such images that have been perturbed using the DeepFool method [18] for the VGG-16 network. The figure clearly shows that even though the label is changed by the added perturbation, the proposed approach is still able to correctly localize the object regions in both cases. Note that the explanations provided by the gradient based methods (e.g. [25]) get affected by the adversarial perturbation. This shows that our approach is robust to images perturbed with adversarial noise to locate the object present in the image.

G. Generic Object Proposal

In this subsection we demonstrate that CNNs trained for object recognition can also act as generic object detectors. Existing algorithms for this task (e.g. [2], [5], [31], [39]) typically provide hundreds of class agnostic proposals, and their performance is evaluated by the average precision and recall measures. While most of them perform very well for large number of proposals, it is more useful to get better metrics at lower number of proposals. Investigating the performances for thousands of proposals is not appropriate since a typical image rarely contains more than a handful of objects. Recent approaches (e.g. [33]) attempt for achieving better

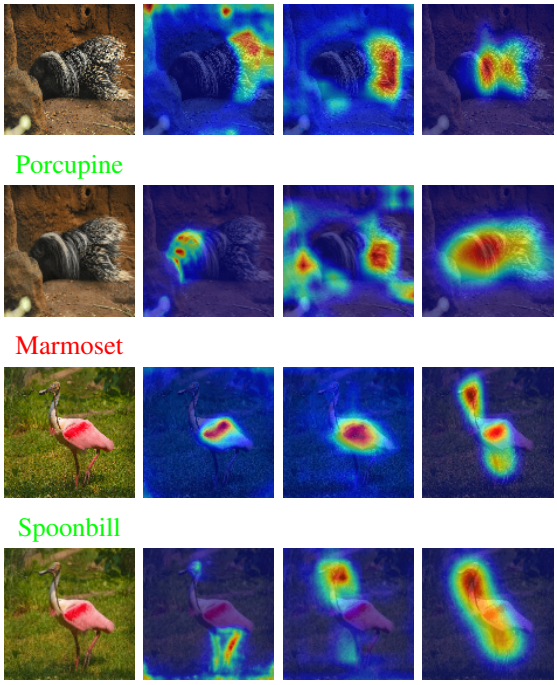


Fig. 9: Visual explanations for sample adversarial images provided by multiple methods. First and third rows show the evidence for the clean samples for which the predicted label is shown in green. Second and fourth rows show the same for the corresponding DeepFool [18] adversaries for which the label is shown in red.

performances at smaller number of proposals. In this section, we take this notion to extreme and investigate the performance of these approaches for ***single*** best proposal. This is because, the proposed method can provide visual explanation for the predicted label and while doing so it can locate the object region using a single proposal. Therefore it is fair to compare our proposal with the best proposal of multiple region proposal algorithms.

Using the proposed approach, we generate object proposals for unseen object categories. We evaluated the models trained over ILSVRC dataset on the PASCAL VOC-2007 [8] test images. Note that the target categories are different from that of the training dataset and the models are trained for object recognition. We pass each image in the test set through the CNN and obtain a bounding box (for the predicted label) as explained in IV-A. This proposal is compared with the ground truth bounding box of the image and if the IoU is more than 0.5, it is considered a true positive. We then measure the performance in terms of the mean average recall and precision per class as done in the PASCAL benchmark [8] and [4].

Table III shows the performance of the proposed approach for single proposal and compares it against well known object proposal approaches and other CNN based visualization methods discussed above. For STL [4] the numbers were obtained from their paper and for other CNN based approaches we used GoogLeNet [30] as the underlying CNN. The objective of this experiment is to demonstrate the ability of CNNs as generic

TABLE III: The performance of different methods for Generic Object Proposal generation on the PASCAL VOC-2007 test set. Note that the methods are divided into CNN based and non-CNN based also the proposed method outperforms all the methods along with backprop [27] method. All the CNN based works except [20] use the GoogLeNet [30] and [15] uses a ResNet [10] architecture to compute the metrics. In spite of working with the best CNN, [20] performs on par with our approach (denoted with $*$).

Type	Method	mRecall	mPrecision
Non-CNN	Selective Search	0.10	0.14
	EdgeBoxes	0.18	0.26
	MCG	0.17	0.25
	BING	0.18	0.25
CNN	Backprop	**0.32**	0.36
	CAM	0.30	0.33
	cMWP	0.23	0.26
	Grad-CAM	0.18	0.21
	STL-WL	0.23	0.31
	Deep Mask [20]	0.29*	**0.38***
	Ours	**0.32**	0.36

object detectors via localizing evidence for the prediction. The proposed approach outperforms all the non-CNN based methods by large margin and performs better than all the CNN based methods except the Backprop [27] and DeepMask [20] methods, which perform equally. Note that [20], in spite of using a strong net (ResNet) and training procedure to predict a class agnostic segmentation, performs comparable to our method.

V. Conclusion

We propose an unfolding approach to trace the evidence for a given neuron activation, in the preceding layers. Based on this, a novel visualization technique, CNN-fixations is presented to highlight the image locations that are responsible for the predicted label. High resolution and discriminative localization maps are computed from these locations. The proposed approach is computationally very efficient which unlike other existing approaches doesn't require to compute either the gradients or the prediction differences. Our method effectively exploits the feature dependencies that evolve out of the end-to-end training process. As a result only a single forward pass is sufficient to provide a faithful visual explanation for the predicted label.

We also demonstrate that our approach enables interesting set of applications. Furthermore, in cases of erroneous predictions, the proposed approach offers visual explanations to make the CNN models more transparent and help improve the training process and annotation procedure.

VI. Acknowledgements

The authors thank Suraj Srinivas for having helpful discussions while conducting this research.

References

[1] M. Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[2] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *IEEE Computer Vision and Pattern Recognition, (CVPR)*, 2014.

[3] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7), 2015.

[4] L. Bazzani, A. Bergamo, D. Anguelov, and L. Torresani. Self-taught object localization with deep networks. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016.

[5] M. M. Cheng, Z. Zhang, W. Y. Lin, and P. H. S. Torr. BING: Binarized normed gradients for objectness estimation at 300fps. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2014.

[6] H. Cholakkal, J. Johnson, and D. Rajan. Backtracking ScSPM image classifier for weakly supervised top-down saliency. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[7] M. Eitz, J. Hays, and M. Alexa. How do humans sketch objects? *ACM Transactions on Graphics (TOG)*, 31(4), 2012.

[8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results.

[9] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition (CVPR)*, 2016.

[11] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8), 1997.

[12] G. Huang and Z. Liu. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[13] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, 2014.

[14] A. Karpathy and F. Li. Deep visual-semantic alignments for generating image descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

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

[16] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision (ECCV)*, 2014.

[17] M. Marszalek and C. Schmid. Accurate object localization with shape masks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[18] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[19] K. R. Mopuri, U. Garg, and R. V. Babu. Fast feature fool: A data independent approach to universal adversarial perturbations. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2017.

[20] P. O. Pinheiro, R. Collobert, and P. Dollár. Learning to segment object candidates. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[21] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*. 2015.

[22] M. Robnik-Šikonja and I. Kononenko. Explaining classifications for individual instances. *IEEE Transactions on Knowledge and Data Engineering*, 20(5), 2008.

[23] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3), 2015.

[24] R. K. Sarvadevabhatla, J. Kundu, and R. V. Babu. Enabling my robot to play pictionary: Recurrent neural networks for sketch recognition. In *ACM Conference on Multimedia*, 2016.

[25] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[26] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

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

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

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

[30] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE conference on computer vision and pattern recognition (CVPR)*, 2015.

[31] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *International journal of computer vision (IJCV)*, 104(2), 2013.

[32] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: Lessons learned from the 2015 mscoco image captioning challenge. *IEEE transactions on pattern analysis and machine intelligence (TPAMI)*, 39(4), 2017.

[33] B. Yang, J. Yan, Z. Lei, and S. Li. Craft objects from images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[34] Q. Yu, Y. Yang, Y. Song, T. Xiang, and T. M. Hospedales. Sketch-a-net that beats humans. In *British Machine Vision Conference (BMVC)*, 2015.

[35] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, 2014.

[36] J. Zhang, Z. Lin, S. X. Brandt, Jonathan, and S. Sclaroff. Top-down neural attention by excitation backprop. In *European Conference on Computer Vision (ECCV)*, 2016.

[37] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2016.

[38] L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling. Visualizing deep neural network decisions: Prediction difference analysis. In *International Conference on Learning Representations (ICLR)*, 2017.

[39] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision (ECCV)*, 2014.