How You See Me: Understanding Convolutional Neural Networks

Rohit Gandikota*
Dept. of Avionics
Indian Institute of Space science and Technology
Trivandrum, Kerala, India
grohit0@gmail.com

Deepak Mishra
Dept. of Avionics
Indian Institute of Space science and Technology
Trivandrum, Kerala, India
deepak.mishra@iist.ac.in

***Abstract*—Convolutional Neural networks(CNN) are one of the most powerful tools in the present era of science. There has been a lot of research done to improve their performance and robustness while their internal working was left unexplored to much extent. They are often defined as black boxes that can map non-linear data effectively. This paper answers the question, "How does a CNN look at an image?". Visual results are also provided to strongly support the proposed method.**

The proposed algorithm exploits the basic math behind CNN to backtrack the important pixels. This is a generic approach which can be applied to any architecture of a neural network. This doesn't require any additional training or architectural changes. In literature, few attempts have been made to explain how learning happens in CNN internally, by exploiting the convolution filter maps. This is a simple algorithm as it does not involve any cost functions, filter exploitation, gradient calculations or probability scores. Further, we demonstrate that the proposed scheme can be used in some important computer vision tasks such as object detection, salient region proposal, etc.

***Index Terms*—Convolutional Neural Network, activation maps, max pooling, convolution, filters, backtracking, pixels**

I. Introduction

Convolution Neural Networks(CNN) have been revolutionizing the area of computer vision with their outstanding performance in vision tasks. They are non-linear functions which are designed to model a human eye. Acknowledging their importance, research has been done to improve their performance. Over time, the accuracy of the system increased and so did the complexity behind it's working. Many complex architectures are being introduced to improve their performance. For example, starting from AlexNet [1] with 8 layers, then came ZFNet [2] and VGGNet [3]. Present Resnet [4] has hundreds of layers with 50 times a lesser number of parameters compared to AlexNet. These models offer less evidence on how they work internally and achieve such extraordinary results.

One approach for understanding CNN is by exploiting the feature activation maps of the filters in the network. Another way is by proposing the regions that the CNN is looking at, in the image. The common motivation behind these methods are to propose the regions in the image that corresponds to the CNN's output of recognizing objects in the image. Understanding the working of CNN's are important because

- We can guide the network's training more accurately when we can visualize the learning at each epoch of the training.
- A visual evidence can be given to explain any alternate identification (i.e a different classification from the ground truth). Some times a network can identify or move it's attention more towards another less important object in an image. This can be visualized also.
- Development in the area of neurology can happen by understanding from the CNN's working which is nothing but a human eye model.
- Many other applications like salient region proposal, object detection, etc. can be improved and simplified. Self driving cars can use this method so as to understand what humans see before they take certain decisions on road.

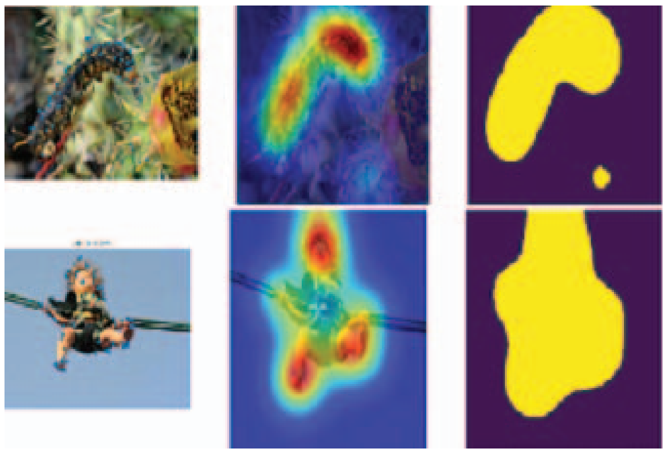

Fig. 1. Results from MSRA-B dataset. First column shows the important pixels, second column shows the attention regions and the final column depicts the saliency region in the images.

The proposed algorithm fundamentally exploits the basic working of the CNN to backtrack and find the important pixels in the image that the CNN observes to make a decision. We try to unroll the forward pass in the CNN after passing an image by reverse engineering each layer's functioning. Given a node,

*Rohit Gandikota is an undergraduate student from Avionics department, IIST, Kerala, India. **He is student ambassador for Intel AI developer program.**

we try to find the nodes from the previous layer that activates it the most. This has been done from the final output layer till the input layer to get the pixel level information. So given an image, we pass it through a CNN that is trained to classify and then we simply backtrack the nodes in a tree fashion from the output till the input layer.

The rest of this paper is organized as follows: section 2 discusses the existing works that are relevant for the work, section 3 presents the proposed method in detail, section 4 explains the potency of our method on salient region detection empirically, section 5 talks about the future work and improvements we are planning on and finally section 6 concludes the work presented.

II. Previous Work

In this section, previous works in this area of research are discussed. This work is mainly motivated by a very dynamic algorithm called Viterbi Algorithm [11]. Viterbi algorithm is for finding the optimal sequence of hidden states. Given an observation sequence and a Hidden Markov Model(HMM), the algorithm returns the state path through the HMM that has a maximum likelihood for the observations. Viterbi algorithm is similar to a forward pass, except has one component: back-pointers. The reason is that while the forward algorithm needs to produce an observation likelihood, the Viterbi algorithm must produce a probability and the most likely state sequence. We compute this sequence by keeping a check on the path of hidden states that led to each state and then at the end back-tracking the best path to the beginning (the Viterbi backtrack).

Few attempts were made to understand CNN previously, most of them are gradient based. They find out the image regions which can improve the probability score for an output class. The work presented in [5] measures the sensitivity of the classification score for a class by tweaking the pixel values. They compute partial derivative of the score in the pixel space and visualize them. Another approach was visualizing using deconvolution, as shown in [6]. The deconvolution approach visualizes the activation maps(filter maps) learned by the network.

Some other works like [7], [8], [10] were done by taking relevance score for each feature with respect to a class. The idea was to see how the output was effected if a feature is dropped. The importance of the feature was based on the change in the score.

III. Proposed Method

This section discusses the proposed method and it's algorithm in detail. Since we attempt to unravel the functioning of convolutional neural network, understanding it is vital. Every CNN is made of common blocks in the form of convolution, pooling and fully connected. In this section, we explain how we backtracked through these layers to determine discriminative image regions in the form of important pixels.

There is no need for any extra training for this, we used a pre-trained VGG19 network on Imagenet dataset. Our algorithm exploits the CNN working during testing time. Backtracking fully connected, convolution and pooling layers are described in detail below

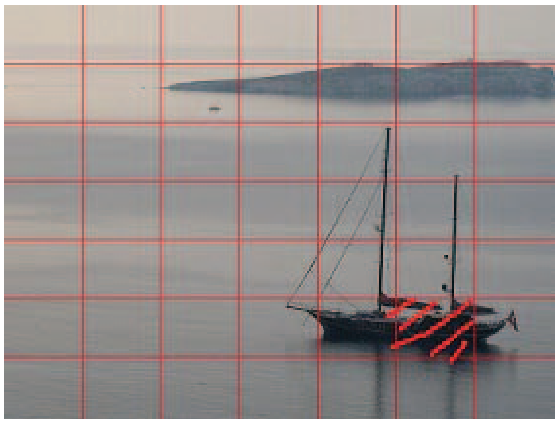

Fig. 2. After backtracking through all the fully connected layers, we end up with nodes in 7x7x512 dimensional space. Projecting the top responsible node in that dimension onto the input image of size 224x224 results in the above visualization. The image has been split into equal 7x7 boxes and box with same co-ordinates as the node is selected to visualize the important region.

A. Backtracking Fully Connected Layers

Typically for recognition, the final layer will be fully connected with a soft-max activation. We maintained a two memory vector, $M1$ for storing the important node locations in the succeeding layer and $M2$ to start storing the locations of important nodes in the present layer. For every node N in the memory vector $M1$, we track the nodes in the present layer that are most responsible to activate it. So for example in the last layer since we are interested in the predicted label, our memory vector $M1$ has only one node corresponding to the label . We start storing all the nodes in the fc2 layer of VGG19 that are most responsible(i.e. top $'n'$ number of nodes that contribute) for the activation of the node in memory vector $M2$. Fig. 4 shows the process of backtracking through fc layers.

For the of understanding *Algorithm 1*, let us assume that the memory vector of succeeding layer has m number of nodes. And since the fc layers are 1 dimensional, we stored the locations by just a single number. But for the first fc layer whose previous layer would be a spatial layer, we need to store the location in a tuple of 3 dimensions $[filter, x, y]$. Further, we backtrack through convolution and max-pooling layers till the input image to plot important pixels on the image.

As shown in Fig.2, this tuple can be visualized over an image by plotting the top responsible node on the image as a box. This visualization is valid as the layers before the final fc layer are all spatial and hence the spatial location of the image is restored even in the filtered activations. Hence, to visualize the tuple, we split the image into equal sized grids of the size of the final spatial layer's activation and extrapolate the point onto the grid (by assuming each grid corresponds to a point). This visualtion result is shown in Fig.2 and here we only considered the top activation alone. This visualization helps in providing a tentative region proposal or salient object detection proposal.

Algorithm 1: Backtracking through fully connected layers

```
1  M1 : Memory vector that has node locations from higher layer
2  M2 : Empty memory vector to store locations of present layer
3  m: Number of nodes in M1 vector
4  for i=1:m do
5      W,b=weights and bias from this layer to node M1[i]
6      A = Activations in the present layer
7      array = W*A + b
8      M2.append(arg(array > 0))
9  end
10 M1 <- M2
```

B. Backtracking through Convolution Layers

As discussed above upon reaching the first fc layer while backtracking, the next layer would be convolution or pooling layers. This subsection would talk about how we back tracked through these layers. Note that from here, all the layers have a 2D(in case of pooling) or 3D(in case of convolution) receptive fields. Also we have stored our nodes in the first fc layer in 3d tuple.

As shown in Fig. 5, for each important node in the present layer l, we extract the receptive field from the previous layer $l-1$. Now we calculate the dot product between weights and check for the feature in $l-1$ layer that has maximum activation. Later to get the x and y co-ordinates, we took the node with maximum activation in the feature map that we have extracted earlier. Note that once the dot product is calculated, the result would be the same shape as receptive field in the previous layer. We calculated the sum over x and y axes to find out the maximum activating feature. Similarly after we extracted the feature, we just took the maximum activating node in that feature. This location is stored in a memory vector that is sent back to above layers. Algorithm 2 explains the above mentioned process

C. Backtracking through Max pooling Layers

As discussed earlier, Max pooling has a 2D receptive field. As explained in *Algorithm 3* and visualized in Fig.6, for every important node in the present layer, we extract the receptive field in the previous layer and find the node with maximum activation.

Note that we have basically unwrapped the working of all the layers to backtrack the important pixels in the image. So after passing the image through the CNN, we extract the activation maps and go backwards unwrapping all the layer functions and finally reach the input layers. That after visualizing gives the important pixels in the image, as shown in Fig.7.

Algorithm 2: Backtracking Convolution layers

```
1  M1 : Memory vector that has node locations from higher layer
2  M2 : Empty memory vector to store locations of present layer
3  for i=1:m do
4      W,b=weights and bias from this layer to node M1[i]
5      A = Activations in the present layer
6      array = W*A + b
7      C = sum(array,axes=x,y)
8      channel = arg(C > 0)
9      x,y = unravel_index(argmax(array[channel]))
10     M2.append([channel,x,y])
11 end
```

Algorithm 3: Backtracking Maxpooling layers

```
1  $rf(n)$ = function that extracts receptive field of the node $n$ M1 : Memory vector that has node locations from higher layer
2  M2 : Empty memory vector to store locations of present layer
3  for i=1:m do
4      A = Activations in the present layer
5      array = $rf(M1[i])$
6      C = argmax(array)
7      channel = M1[i][0]
8      x,y = unravel_index(C)
9      M2.append([channel,x,y])
10 end
```

D. Implementation

We have used **Intel Xeon Gold 6128** processors via Intel DevCloud for this work. This enabled us in reducing the test time by a factor of 10 over an Intel i7 processor. The computational complexity on this processor in terms of time taken per epoch is around 1 sec/epoch. Input size of 128x128x3 was used as required by the pretrained model. It is to be noted that there is no training involved and hence the complexity mentioned is with respect to test time.

IV. Applications

There are many applications associated with this method. For example, it can be used as

- a visual tool for attention regions of CNN in the image.
- an object detection algorithm by drawing bounding boxes around the extreme lying important pixels.
- a salient region proposing algorithm by applying a gaussian backdrop over important pixels.
- a tool to better understand CNN and for better training.

Note that the simplicity in the method makes the computational complexity very less for all the above-mentioned applications. We have done some detailed analysis of salient region detection.

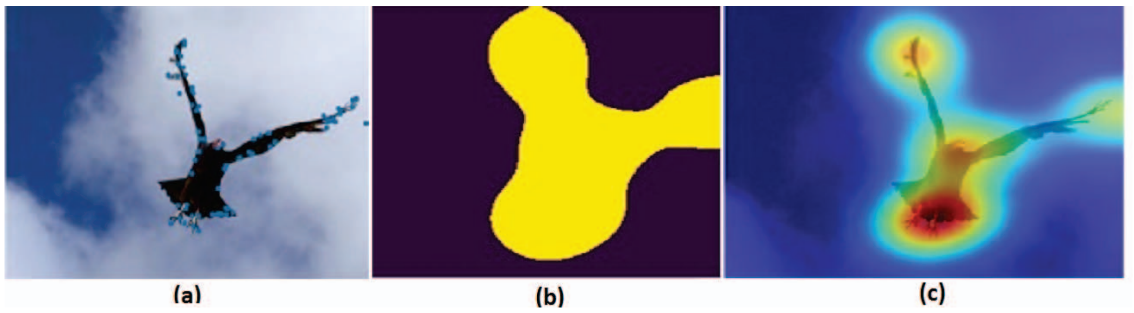

Fig. 3. (a) The important pixels tracked back after a forward pass through a VGG19 network. (b) Saliency map derived from the pixels. (c) Attention region of the image based on the important pixel density. Red being the region with the most attention.

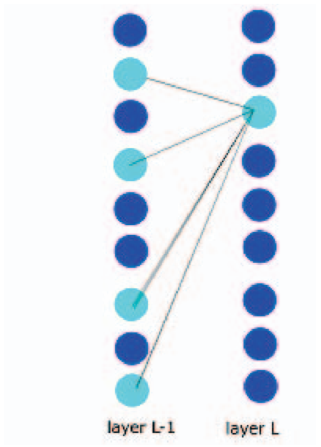

Fig. 4. Backtracking through fully connected layers. The turquoise colored nodes in layer $L-1$ positively activate the node in layer L. These are the nodes that are stored in the memory $M2$

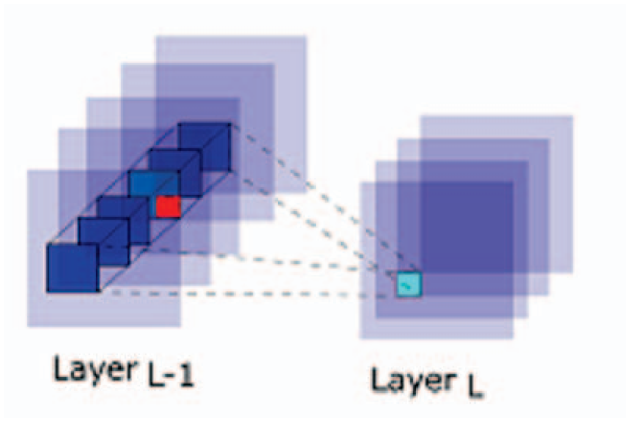

Fig. 5. Backtracking through Convolution layers. The turquoise colored channel in layer $L-1$ is the maximum contributing channel for the activation of turquoise colored node in layer L. Further the red colored node is selected as it is the maximum contributing node from the selected channel. This channel and the co-ordinates of the red colored node are stored in the memory $M2$

A. Salient region detection

After obtaining the important pixels in the image as shown in Fig.3(a), we have drawn Gaussian around each pixel and put a threshold to obtain the salient maps as shown in Fig.3(b). We have run our algorithm on MSRA-B dataset and the results are shown in Table 1. Attention region can also be visualized without thresholding the gaussians. Best values were chosen for the standard deviation of Gaussian and the threshold value after considering several random values. Some interesting observations were seen, like

- Even though the class of the object present in the image is not known to CNN, it looks at the object at meaningful regions.
- This proposed method works good with blurred images also as shown in Fig.8(a).
- The attention region is not just the recognized object, but also some part of the background as shown in Fig.8(b). This explains the precision values to be lower as compared to recall values.

TABLE I
COMPARISON OF SALIENT REGION PROPOSALS BY OUR MODEL ON MSRA-B DATASET COMPARED TO THE PROVIDED GROUND-TRUTH

Accuracy	Precision	Recall	F-Score	IOU
0.98	0.5	0.8	0.7	0.6

V. Future Work

In the future, we would like to extend this algorithm further on to densenet [12] and resnet [4]. We would also like to build an extension to architectures which stores the best activations directly and give a quick response. Also, we would like to explore different applications as this can revolutionize the complexity.

VI. Conclusion

We present a simple generic method to visualize and understand how a CNN looks at an image, by backtracking all the operations of CNN on an image. This method proposes the pixels in an image that are mostly responsible for the CNN's decision/classification. We have also shown that salient regions in an image can be identified using this method. We would explore other applications in the future and also would like to improve this algorithm to make it faster and applicable to recurrent networks. We succeeded in understanding a magnifi-

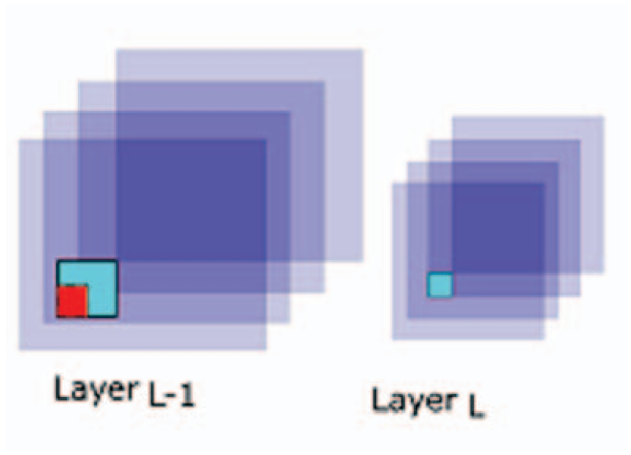

Fig. 6. Backtracking through Max pooling layers. The turquoise colored 2x2 nodal region in layer $L-1$ is the receptive field for max pooling operation of the node highlighted in layer L. Further the red colored node is the maximum contributing node from the field. This channel and the co-ordinates of the red colored node are stored in the memory $M2$

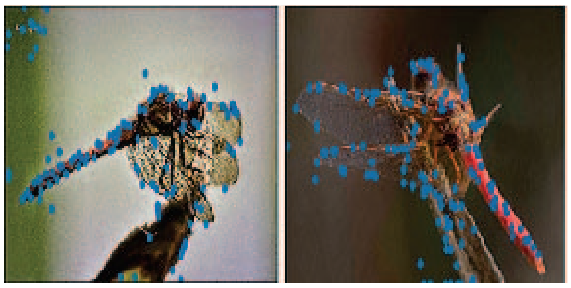

Fig. 7. Results on dragonfly images. There is a pattern followed by the CNN in directing most of it's attention to the head and the tail of a dragonfly.

cent tool's working in a more simpler way and from a different point of view.

Acknowledgment

We would like to express our special thanks of gratitude to Dr. Vineeth.B.S, who gave us the technical support and many valuable suggestions over the course of this project.

Secondly we would also like to thank Intel AI developer program for providing me with an access to their state-of-the-art processor which helped in finishing the project on time. We would also like to thank VR Lab of IIST for accommodating this project by providing high computational resources and also the support from the team.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems (NIPS). 2012.
[2] W.-K. Chen, Linear Networks and Systems (Book style). Belmont, CA: Wadsworth, 1993, pp. 123135.
[3] H. Poor, An Introduction to Signal Detection and Estimation. New York: Springer-Verlag, 1985, ch. 4.
[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015.

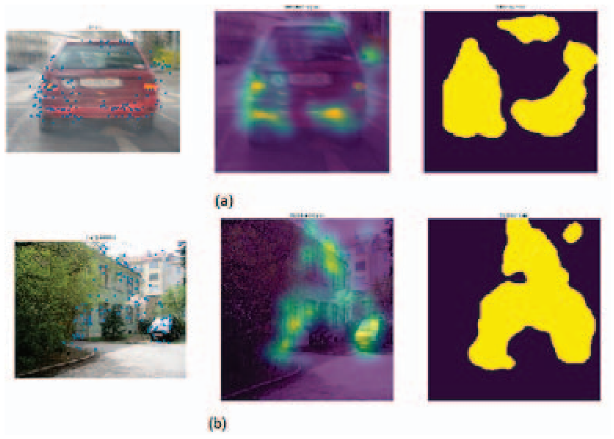

Fig. 8. (a) Results on a blurred image of car. (b) Background is also shown some attention other than the class object car.

[5] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034, 2013.
[6]] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In European Conference on Computer Vision (ECCV), 2014.
[7] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Muller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PloS one, 10(7), 2015.
[8] 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.
[9] 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.
[10] M. Robnik-Sikonja and I. Kononenko. Explaining classifications for individual instances. IEEE Transactions on Knowledge and Data Engineering, 20(5), 2008.
[11] Forney Jr., D.G.: The Viterbi algorithm, Proc. IEEE, 1973, 3, (61), pp. 268278.
[12] Huang, Gao and Liu, Zhuang and van der Maaten, Laurens and Weinberger, Kilian Q. Densely connected convolutional networks.Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.