

Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps

Karen Simonyan, Andrea Vedaldi, Andrew
Zisserman

Presenter: Ankur Garg

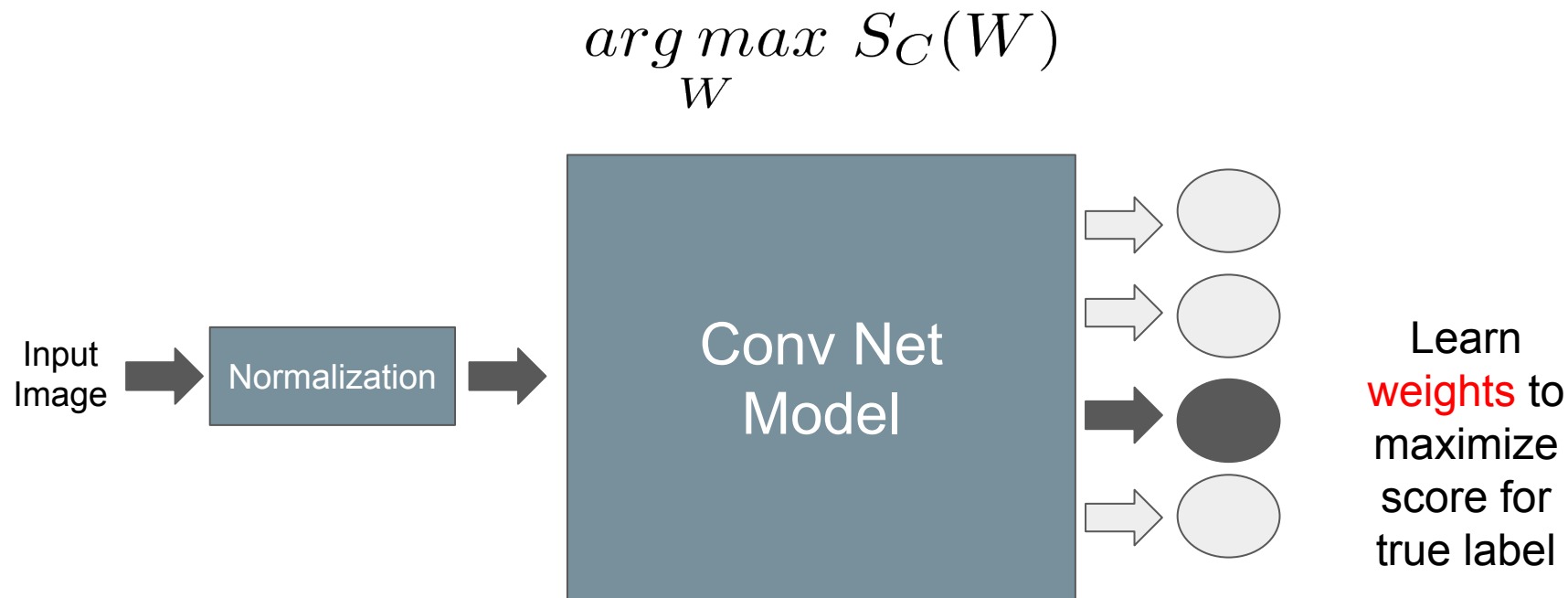
Contributions

1. Obtain **Understandable Visualisations** of Conv Nets.
2. Image-specific class **saliency map**
 - a. Can be used weakly supervised object localization

Contributions

1. Obtain **Understandable Visualisations** of Conv Nets.
2. Image-specific class saliency map
 - a. Can be used weakly supervised object localization

Class Model Visualization



Class Model Visualization

$$\arg \max_I (S_C(I) - \lambda \|I\|_2^2)$$



Results: Class Model Visualization

- Key aspects of images in a class are captured



Bell Pepper



Dumbbell

Contributions

1. Obtain Understandable Visualisations of Conv Nets.
2. Image-specific class **saliency map**
 - a. Can be used weakly supervised object localization

Image-Specific Class Saliency Visualisation

Linear Score Model

$$S_C = w_C^T I + b_C$$

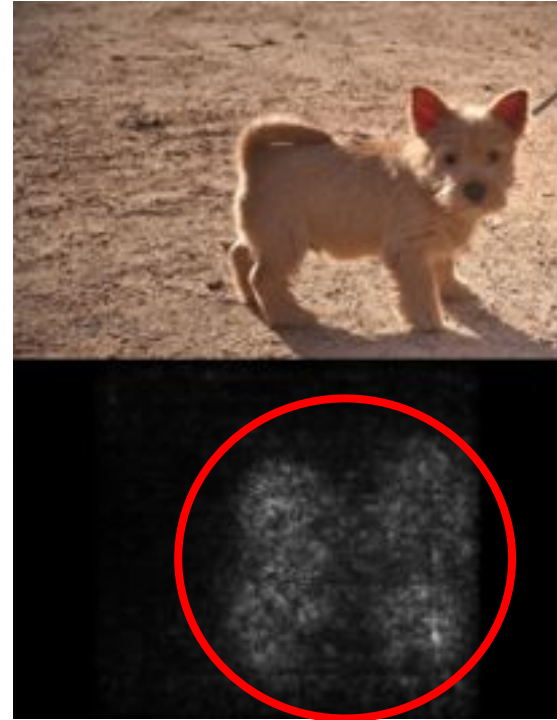
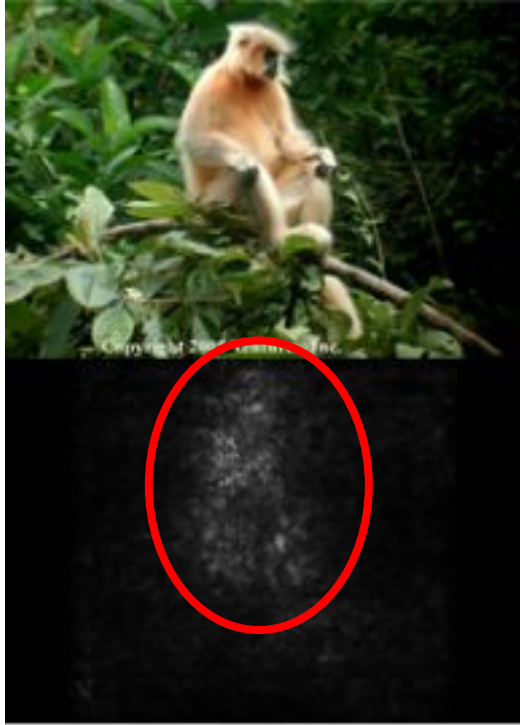
Taylor approximation
of Classification
Function

$$S_C \approx w^T I + b$$

Find w through **single
backpropagation** step

$$w = \left. \frac{\partial S_c}{\partial I} \right|_{I_0}$$

Results: Image-Specific Class Saliency Visualisation



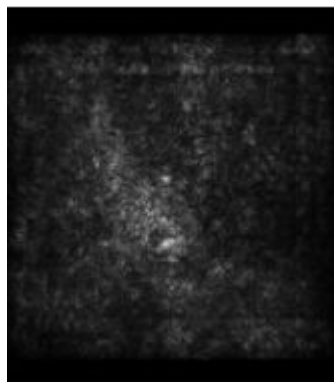
Contributions

1. Obtain Understandable Visualisations of Conv Nets.
2. Image-specific class saliency map
 - a. Can be used weakly supervised object localization

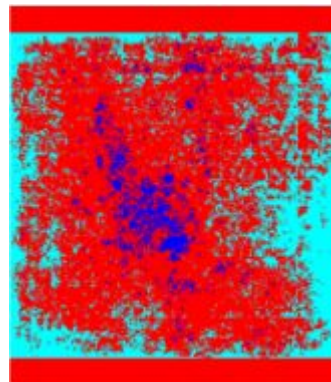
Weakly Supervised Object Localisation



Input Image



Saliency Map



Graph Cut Color
Segmentation Output



Foreground
Segmentation Mask

Results: Weakly Supervised Object Localisation

- ILSVRC-2013 Localisation Challenge
- Top 5 Error: 46.4%
 - Best submission: 29.9% (Fully Supervised Model)
- Authors' previous full supervised submission achieved 50%

Contributions

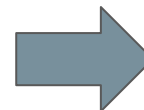
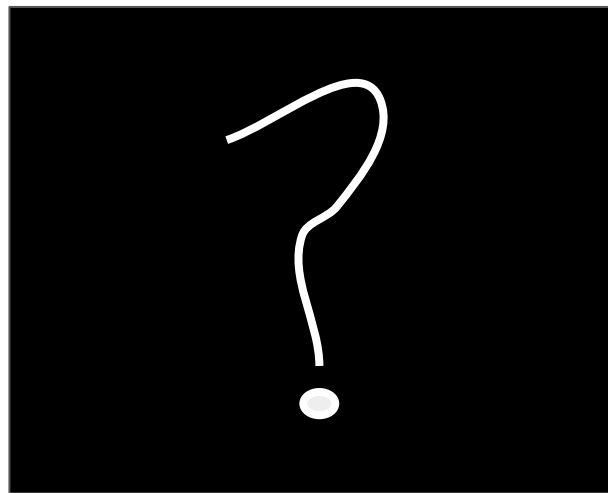
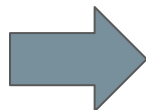
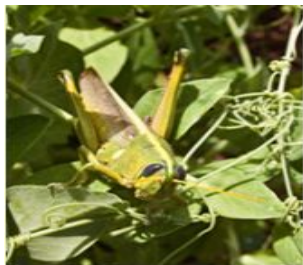
1. Obtain Understandable Visualisations of Conv Nets.
2. Image-specific class saliency map
 - a. Can be used weakly supervised object localization
3. **Generalization** to existing deconvolutional network reconstruction procedure

Relation to Deconvolutional Networks

- Flipped Kernel in Backpropagation matches exactly to reconstruction layer
- Argmax during backpropagation of a max pool layer corresponds to max-pool in reconstruction

Discussion of Pros

Decoding the Black Box

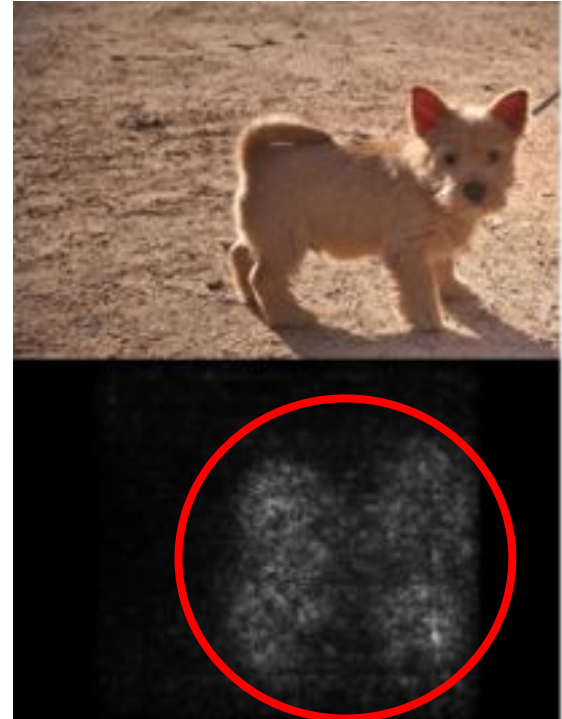


Grasshopper

Intuitive Visual Results



Dumbbell

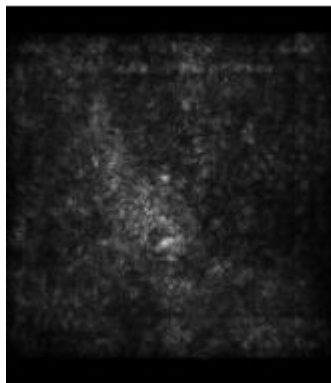


Importance of Image Saliency Maps

- Extending to Object Localisation shows usefulness of Saliency Maps
- Opened up research avenues for incorporating saliency maps.



Input Image



Saliency Map



Foreground Segmentation Mask

Easy to Understand & Implement

- Saliency Map extraction requires only a single backpropagation step
- Techniques easily translatable to code
 - <https://raghakot.github.io/keras-vis/visualizations/saliency/>

```
39 def visualize(data, padsize=1, padval=0):
40     data = copy.deepcopy(data)
41     data -= data.min()
42     data /= data.max()
43
44     # Force the number of filters to be square
45     m = int(np.ceil(np.sqrt(data.shape[0])))
46     padding = ((0, m ** 2 - data.shape[0]), (0, padsize), (0, 0),) * (data.ndim - 3)
47     data = np.pad(data, padding, mode='constant', constant_values=(padval, padval))
48
49     # Tile the filters into an image
50     data = data.reshape((m, m) + data.shape[1:]).transpose((0, 2, 1, 3) + tuple(range(4, data.ndim + 1)))
51     data = data.reshape((m * data.shape[1], m * data.shape[1]) + data.shape[1:])
52
53     plt.imshow(data)
54     plt.show(block=False)
55
56     return data
57
58 for i in range(N_ITERATIONS):
59     # Perform forward pass
60     aff = net.forward(data=caffe_data, label=caffe_label)
61     # Perform backward pass
62     aff = net.backward()
63
64     # Perform gradient ascent and update caffe_data
65     diff = buf['data']
66     caffe_data -= learning_rate * caffe_data * diff
67     print i
68
69     forwardOutput = net.forward(data=caffe_data)
70     prediction = net.predict([input_image])
71
72     # Find the saliency map as described in the paper. Normalize the map and assign it to variable "saliency"
73     saliency = np.abs(diff).sum(axis=0)
74
75     # Visualize the caffe_data using visualize function
76     print i
77     vis = visualize(caffe_data.transpose(0,2,3,1))
78     plt.imshow(vis)
79     plt.savefig('part1.png')
```

Class Model Visualization

```
39 label_index = 201 # Index for cat class
40 caffe_data = np.random.random((1, 3, 227, 227))
41 caffe_label = np.zeros((1, 1000, 1, 1))
42 caffe_label[0, label_index, 0, 0] = 1
43
44 def visualize(data, padsize=1, padval=0):
45     data = copy.deepcopy(data)
46     data -= data.min()
47     data /= data.max()
48
49     # Force the number of filters to be square
50     m = int(np.ceil(np.sqrt(data.shape[0])))
51     padding = ((0, m ** 2 - data.shape[0]), (0, padsize), (0, 0),) * (data.ndim - 3)
52     data = np.pad(data, padding, mode='constant', constant_values=(padval, padval))
53
54     # Tile the filters into an image
55     data = data.reshape((m, m) + data.shape[1:]).transpose((0, 2, 1, 3) + tuple(range(4, data.ndim + 1)))
56     data = data.reshape((m * data.shape[1], m * data.shape[1]) + data.shape[1:])
57
58     plt.imshow(data)
59     plt.show(block=False)
60
61     return data
62
63 # Perform a forward pass with the data as the input image
64 pred = net.predict([input_image])
65
66 # Perform a backward pass for the cat class (201)
67 buf = net.backward(["net_output[0]": caffe_label])
68 diff = buf['data']
69
70 # Find the saliency map as described in the paper. Normalize the map and assign it to variable "saliency"
71 saliency = np.abs(diff).sum(axis=0)
72
73 # Visualize the saliency map
74 plt.subplot(1,2,1)
75 plt.imshow(saliency, cmap=gray_r)
76 plt.subplot(1,2,2)
77 plt.imshow(net.transformer.deprocess('data', net.blob['data'].data[0]))
78 plt.show()
79 plt.savefig('saliency.png')
```

Saliency Map Extraction

References

- [Visualizing and Understanding Deep Neural Networks](#) by
Matt Zeiler