…EGORY/ADVANCED/)

…ATEGORY/ALGORITHM/)

COMPUTER VISION (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/COMPUTER-VISION/)

DEEP LEARNING (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/DEEP-LEARNING/)

IMAGE (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/IMAGE/)

PYTHON (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/PYTHON-2/)

UNSTRUCTURED DATA (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/UNSTRUCTURED-DATA/)

A Guide to Understanding Convolutional Neural Networks (CNNs) using Visualization

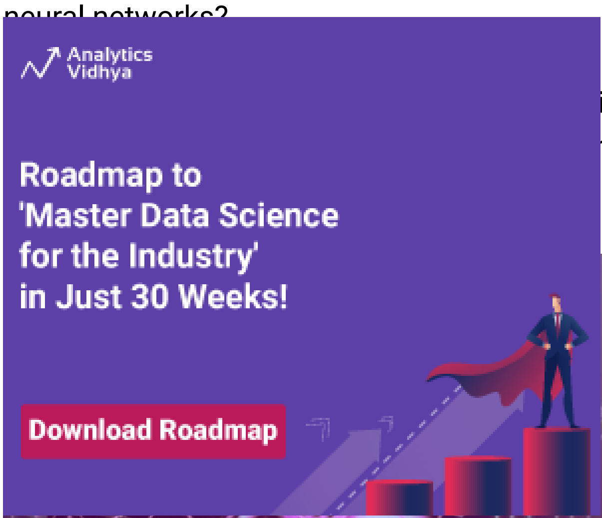
SAURABH PAL (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/AUTHOR/SAURABHPAL97/), MAY 6, 2019 LOGIN TO BOOKMARK THIS ARTICLE …

Article   Video Book

Introduction

"How did your neural network produce this result?" This question has sent many data scientists into a tizzy. It's easy to explain how a simple neural network works, but what happens when you increase the layers 1000x in a computer vision (https://courses.analyticsvidhya.com/courses/computer-vision-using-deep-learning-version2?utm_source=blog&utm_medium=understanding-visualizing-neural-networks) project?

Our clients or end users require interpretability – they want to know how our model got to the final result. We can't take a pen and paper to explain how a deep neural network works. So how do we shed this "black box" image of neural networks?



...ith visualizing the different features of a neural network is ...e dealing with a convolutional neural network (CNN) trained on

(https://cdn.analyticsvidhya.com/wp-content/uploads/2019/05/man-nueral-network-red-.jpg)

In this article, we will look at different techniques for visualizing convolutional neural networks. Additionally, we will also work on extracting insights from these visualizations for tuning our CNN model.

Note: This article assumes you have a basic understanding of Neural Networks and Convolutional Neural Networks. Below are three helpful articles to brush up or get started with this topic:

- *A Comprehensive Tutorial to learn Convolutional Neural Networks from Scratch* (https://www.analyticsvidhya.com/blog/2018/12/guide-convolutional-neural-network-cnn/?utm_source=blog&utm_medium=understanding-visualizing-neural-networks)
- *An Introductory Guide to Deep Learning and Neural Networks* (https://www.analyticsvidhya.com/blog/2018/10/introduction-neural-networks-deep-learning/?utm_source=blog&utm_medium=understanding-visualizing-neural-networks)
- *Fundamentals of Deep Learning – Starting with Artificial Neural Network* (https://www.analyticsvidhya.com/blog/2016/03/introduction-deep-learning-fundamentals-neural-networks/?utm_source=blog&utm_medium=understanding-visualizing-neural-networks)

You can also learn CNNs in a step-by-step manner by enrolling in this free course: Convolutional Neural Networks (CNN) from Scratch (https://courses.analyticsvidhya.com/courses/convolutional-neural-networks-cnn-from-scratch?utm_source=blog&utm_medium=understanding-visualizing-neural-networks)

Table Of Contents

… Decode Neural Networks?

…s to understand how a neural network works, so why turn to the off-beaten path of visualization?

…e. Consider a project where we need to classify images of animals, …itively, we can differentiate between these animals using the image …

…ts. The majority of the snow leopard images will have snow in the …d images will have a sprawling desert.

Here's the problem – the model will start classifying snow versus desert images. So, how do we make sure our model has correctly learned the distinguishing features between these two leopard types? The answer lies in the form of visualization.

> *Visualization helps us see what features are guiding the model's decision for classifying an image.*

There are multiple ways to visualize a model, and we will try to implement some of them in this article.

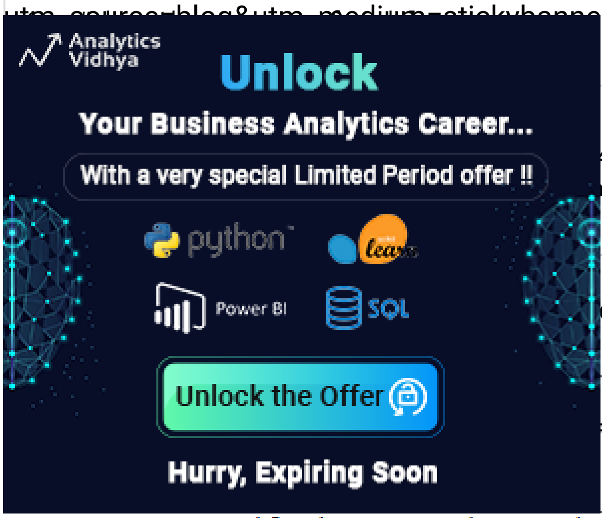
Setting up the Model Architecture

I believe the best way of learning is by coding the concept. Hence, this is a very hands-on guide and I'm going to dive into the Python code straight away.

pretrained weights on the [ImageNet dataset (http://image-](http://image-) port the model into our program and understand its architecture.

the *'model.summary()'* function in Keras. *This is a very important* . We need to make sure the input and output shapes match our odel summary.

```
thus include_top=True
ude_top=True)
```

4ad31f/raw/bd6eb3c750fa6e7f41a306f8a0dba8b72125ca00/Importing%20Model.py) pal97/20baa3feab2a86055e591ee9654ad31f#file-importing-model-py) hosted with

e above code:

	Output Shape	Param #
	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808

```
block4_pool (MaxPooling2D)    (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)         (None, 14, 14, 512)       2359808
_________________________________________________________________
                              (None, 14, 14, 512)       2359808
_________________________________________________________________
                              (None, 14, 14, 512)       2359808
_________________________________________________________________
                         g2D) (None, 7, 7, 512)         0
_________________________________________________________________
                              (None, 25088)             0
_________________________________________________________________
                              (None, 4096)              102764544
_________________________________________________________________
                              (None, 4096)              16781312
_________________________________________________________________
                              (None, 1000)              4097000
=================================================================
Total params: 138,357,544
Trainable params: 138,357,544
                         0
```


along with the number of trainable parameters at every layer. I want ne above output to understand what we have at hand.

re training only a subset of the model layers (feature e the model summary and ensure that the number of atches the layers that we do not want to train.

Also, we can use the total number of trainable parameters to check whether our GPU will be able to allocate sufficient memory for training the model. That's a familiar challenge for most of us working on our personal machines!

Accessing Individual Layers

Now that we know how to get the overall architecture of a model, let's dive deeper and try to explore individual layers.

It's actually fairly easy to access the individual layers of a Keras model and extract the parameters associated with each layer. This includes the layer weights and other information like the number of filters.

Now, we will create dictionaries that map the layer name to its corresponding characteristics and layer weights:

```
#creating a mapping of layer name ot layer details
#we will create a dictionary layers_info which maps a layer name to its charcteristics
layers_info = {}
for i in model.layers:
    layers_info[i.name] = i.get_config()

#here the layer_weights dictionary will map every layer_name to its corresponding weights
```

```
layer_weights = {}
for i in model.layers:
    layer_weights[i.name] = i.get_weights()
```


42c248da5b1593222b/raw/3261092f38f1f74d0027e873fa4a9ab6d7e7c27a/Layers.py)
b2d3a26749b42c248da5b1593222b#file-layers-py) hosted with ❤ by GitHub

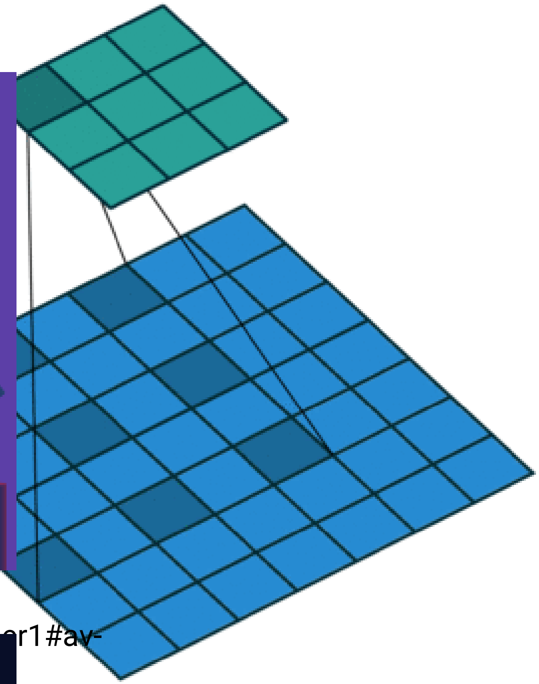
ich consists of different parameters of the *block5_conv1* layer:


```
 'trainable': True,
 'filters': 512,
...
    'class_name': 'VarianceScaling',
...
    'mode': 'fan_avg',
    'distribution': 'uniform',
    'seed': None}},
 'bias_initializer': {'class_name': 'Zeros', 'config': {}},
 'kernel_regularizer': None,
 'bias_regularizer': None,
 'activity_regularizer': None,
 'kernel_constraint': None,
 'bias_constraint': None}
```

Did you notice that the trainable parameter for our layer '*block5_conv1*' is true? This means that we can update the layer weights by training the model further.

Visualizing the Building Blocks of CNNs – Filters

Filters are the basic building blocks of any Convolutional Neural Network (https://courses.analyticsvidhya.com/courses/convolutional-neural-networks-cnn-from-scratch?utm_source=blog&utm_medium=understanding-visualizing-neural-networks). Different filters extract different kinds of features from an image. The below GIF illustrates this point really well:



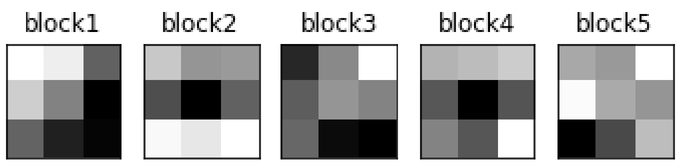
/uploads/2019/04/filter.gif)

omposed of multiple filters. Check out the output we generated in the
onsists of 512 filters. Makes sense, right?

al layer of every VGG16 block:

```
#plot the filters
fig,ax = plt.subplots(nrows=1,ncols=5)
for i in range(5):
    ax[i].imshow(layers[layer_ids[i]].get_weights()[0][:,:,:,0][:,:,0],cmap='gray')
    ax[i].set_title('block'+str(i+1))
    ax[i].set_xticks([])
    ax[i].set_yticks([])
```

iew raw
https://gist.github.com/saurabhpal97/8e6c484546beeb167363bc690fb3ca41/raw/a4b3a681f8cbbae724603685f1b605c3d18adccd/Filters.py)
Filters.py (https://gist.github.com/saurabhpal97/8e6c484546beeb167363bc690fb3ca41#file-filters-py) hosted with ❤ by GitHub
(https://github.com)



(https://cdn.analyticsvidhya.com/wp-content/uploads/2019/04/Filters.png)

We can see the filters of different layers in the above output. **All the filters are of the same shape since VGG16 uses only 3×3 filters.**

Visualizing what a Model Expects – Activation Maximization

Let's use the image below to understand the concept of activation maximization:

...or the model to identify the elephant? Some major ones I can think of:

... right? Now, let's see what we get when we try to optimize a random

...NN (https://courses.analyticsvidhya.com/courses/convolutional-...e=blog&utm_medium=understanding-visualizing-neural-networks) looks for similar patterns in the output of the previous layer. The activation of a convolutional layer is maximized when the input consists of the pattern that it is looking for.

> *In the activation maximization technique, we update the input to each layer so that the activation maximization loss is minimized.*

How do we do this? We calculate the gradient of the activation loss with respect to the input, and then update the input accordingly:

$$\frac{\partial ActivationMaximizationLoss}{\partial input}$$

(https://cdn.analyticsvidhya.com/wp-content/uploads/2019/04/activation_maximization_gradient.png)

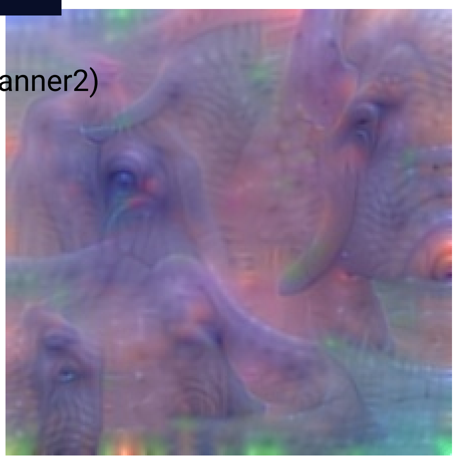
Here's the code for doing this:

```
#importing the required modules
from vis.visualization import visualize_activation
from vis.utils import utils
from keras import activations
```

```
from keras import applications
import matplotlib.pyplot as plt
%matplotlib inline

nected layers also because then we can
category

le_top=True)

r name
e model and name of layer as parameters and return the index of respect
'predictions')
o linear
tivations.linear
#applying modifications to the model
)

_idx,filter_indices=385,max_iter=5000,verbose=True)
```


c5f6499ef3/raw/802891d738aab630a293cc7db15ebc54d631238e/activation_max.py)
al97/158988f112e2e3b6067d25c5f6499ef3#file-activation_max-py) hosted with ❤ by

a random input for the class corresponding to Indian Elephant:



(https://cdn.analyticsvidhya.com/wp-content/uploads/2019/04/elephant_generated.jpg)

From the above image, we can observe that the model expects structures like a tusk, large eyes, and trunk. Now, this information is very important for us to check the sanity of our dataset. For example, let's say that the model was focussing on features like trees or long grass in the background because Indian elephants are generally found in such habitats.

Then, using activation maximization, we can figure out that our dataset is probably not sufficient for the task and we need to add images of elephants in different habitats to our training set.

Visualizing what's Important in the Input- Occlusion Maps

Activation maximization is used to visualize what the model expects in an image. Occlusion maps, on the other hand, help us find out which part of the image is important for the model.

k, we consider a model that classifies cars according to their

ured the above car? Probably not because the part where the company e. That part of the image is clearly important for our classification

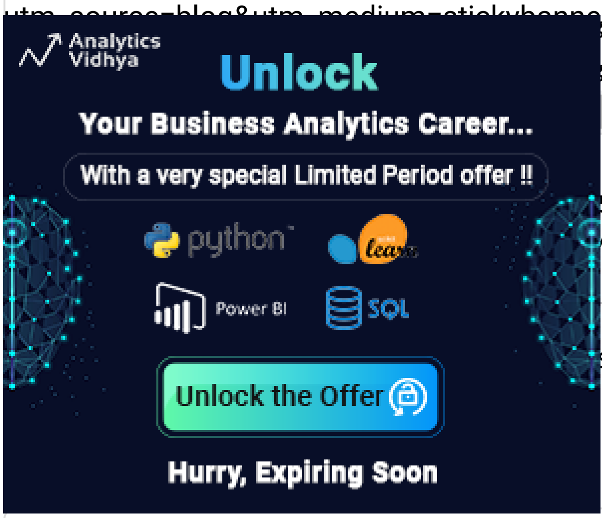
e occlude some part of the image and then calculate its probability of belonging to a class. If the probability decreases, then it means that occluded part of the image is important for the class. Otherwise, it is not important.

Here, we assign the probability as pixel values for every part of the image and then standardize them to generate a heatmap:

```
import numpy as np

from keras.utils import np_utils
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Activation, Conv2D, MaxPooling2D
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping, ModelCheckpoint
from keras.preprocessing.image import ImageDataGenerator
from keras.activations import relu

%matplotlib inline
import matplotlib.pyplot as plt
def iter_occlusion(image, size=8):

    occlusion = np.full((size * 5, size * 5, 1), [0.5], np.float32)
    occlusion_center = np.full((size, size, 1), [0.5], np.float32)
    occlusion_padding = size * 2

    # print('padding...')
```

```
    image_padded = np.pad(image, ( \
                    (occlusion_padding, occlusion_padding), (occlusion_padding, occlusion_padding), (0, 0
                    ), 'constant', constant_values = 0.0)
```


```
...mage.shape[0] + occlusion_padding, size):
...g, image.shape[1] + occlusion_padding, size):
... + occlusion_center.shape[0] + occlusion_padding, \
... + occlusion_center.shape[1] + occlusion_padding] \
...shape[0], x:x + occlusion_center.shape[1]] = occlusion_center
...occlusion_padding, \
...:tmp.shape[0] - occlusion_padding, occlusion_padding:tmp.shape[1] - occ
```

...580a848c8414c/raw/74b5290d2d72253cf50a72cfe66a9ddb616e3c96/occlusion_1.py) .../78d5846646932713ba9580a848c8414c#file-occlusion_1-py) hosted with ❤ by

...*sion* that returns an image with different masked portions.

```
from keras.preprocessing.image import load_img
# load an image from file
image = load_img('car.jpeg', target_size=(224, 224))
plt.imshow(image)
plt.title('ORIGINAL IMAGE')
```

v /gist.github.com/saurabhpal97/78d5846646932713ba9580a848c8414c/raw/74b5290d2d72253cf50a72cfe66a9ddb616e3c96/occlusion_2.py) occlusion_2.py (https://gist.github.com/saurabhpal97/78d5846646932713ba9580a848c8414c#file-occlusion_2-py) hosted with ❤ by GitHub (https://github.com)

Original Image

Now, we'll follow three steps:

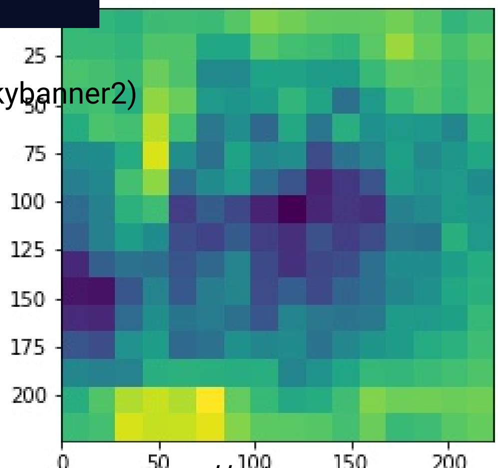
- Preprocess this image
- Calculate the probabilities for different masked portions
- Plot the heatmap

```
from keras.preprocessing.image import img_to_array
from keras.applications.vgg16 import preprocess_input
# convert the image pixels to a numpy array
```



```
]), image.shape[1], image.shape[2]))

tput classes

for n,(x,y,image) in enumerate(iter_occlusion(temp,14)):
ict(image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
```


580a848c8414c/raw/74b5290d2d72253cf50a72cfe66a9ddb616e3c96/occlusion_3.py) /78d5846646932713ba9580a848c8414c#file-occlusion_3-py) hosted with ❤ by



Heatmap

Really interesting. We will now create a mask using the standardized heatmap probabilities and plot it:

```
import skimage.io as io
#creating mask from the standardised heatmap probabilities
mask = heatmap1 < 0.85
mask1 = mask *256
mask = mask.astype(int)
io.imshow(mask,cmap='gray')
```

/gist.github.com/saurabhpal97/78d5846646932713ba9580a848c8414c/raw/74b5290d2d72253cf50a72cfe66a9ddb616e3c96/occlusion_4.py) occlusion_4.py (https://gist.github.com/saurabhpal97/78d5846646932713ba9580a848c8414c#file-occlusion_4-py) hosted with ❤ by GitHub (https://github.com)

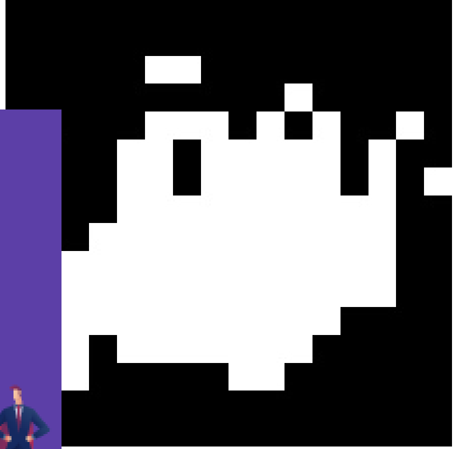


image and plot that as well:

```
2 #read the image
3 image = plt.imread('car_img.jpg')
...R2RGB)
...k = mask)
...R2RGB)
```


...580a848c8414c/raw/74b5290d2d72253cf50a72cfe66a9ddb616e3c96/occlusion_5.py) occlusion_5.py (https://gist.github.com/saurabhpal97/78d584664693271 3ba9580a848c8414c#file-occlusion_5-py) hosted with ❤ by GitHub (https://github.com)

Can you guess why we're seeing only certain parts? That's right – only those parts of the input image that had a significant contribution to its output class probability are visible. That, in a nutshell, is what occlusion maps are all about.

Visualizing the Contribution of Input Features- Saliency Maps

Saliency maps are another visualization technique based on gradients. These maps were introduced in the paper – Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. (https://arxiv.org/pdf/1312.6034v2.pdf)

Saliency maps calculate the effect of every pixel on the output of the model. This involves calculating the gradient of the output with respect to every pixel of the input image.

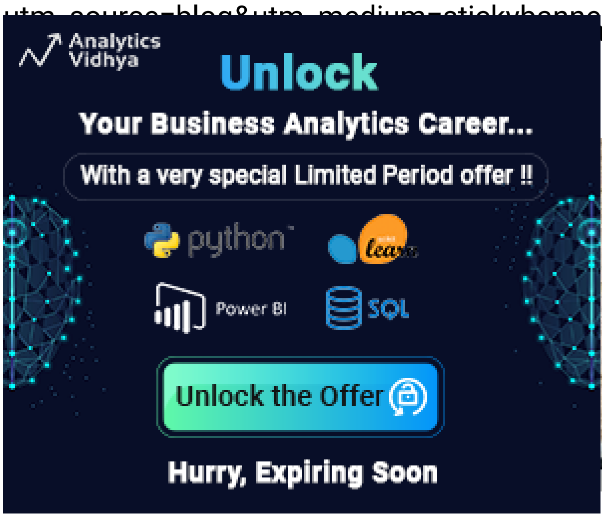
with respect to small changes in the input image pixels. All the changes to the pixel value will increase the output value:

$$\frac{\partial output}{\partial input}$$

/uploads/2019/04/saliency.png)

e as the image (gradient is calculated with respect to every pixel), provide us with the intuition of attention.

ny image. First, we will read the input image using the below code



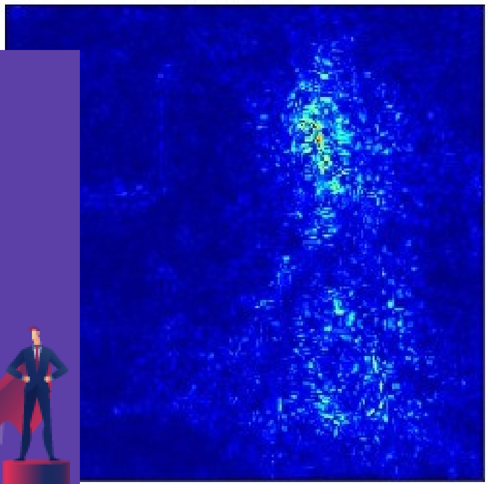
Input Image

Now, we will generate the saliency map for the image using the VGG16 model:

```
# Utility to search for layer index by name.
# Alternatively we can specify this as -1 since it corresponds to the last layer.
layer_idx = utils.find_layer_idx(model, 'predictions')

# Swap softmax with linear
model.layers[layer_idx].activation = activations.linear
model = utils.apply_modifications(model)

#generating saliency map with unguided backprop
grads1 = visualize_saliency(model, layer_idx,filter_indices=None,seed_input=image)
#plotting the unguided saliency map
plt.imshow(grads1,cmap='jet')
```

v raw
ps://gist.github.com/saurabhpal97/6cc03c437b2fee7f8422a545a29ba633/raw/d04675816257e6cfe033e11e90b5f0160be8f4fa/saliency_2.py)
saliency_2.py (https://gist.github.com/saurabhpal97/6cc03c437b2fee7f8422a545a29ba633#file-saliency_2-py) hosted with ❤ by GitHub (https://github.com)

UNGUIDED

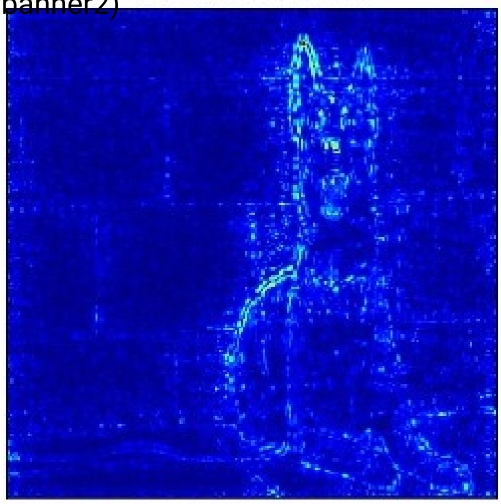


We see that the model focuses more on the facial part of the dog. Now, let's look at the results with guided


```
...kprop
...r_idx,filter_indices=None,seed_input=image,backprop_modifier='guided')
```

...422a545a29ba633/raw/d04675816257e6cfe033e11e90b5f0160be8f4fa/saliency_3.py)
...6cc03c437b2fee7f8422a545a29ba633#file-saliency_3-py) hosted with ❤ by GitHub

GUIDED GRADIENTS



Guided backpropogation truncates all the negative gradients to 0, which means that only the pixels which have a positive influence on the class probability are updated.

Class Activation Maps (Gradient Weighted)

Class activation maps are also a neural network visualization technique based on the idea of weighing the activation maps according to their gradients or their contribution to the output.

The following excerpt from the Grad-CAM paper gives the gist of the technique:

> *...tivation Mapping (Grad-CAM), uses the gradients of ...ts for 'dog' or even a caption), flowing into the final ...uce a coarse localization map highlighting the ...age for predicting the concept.*

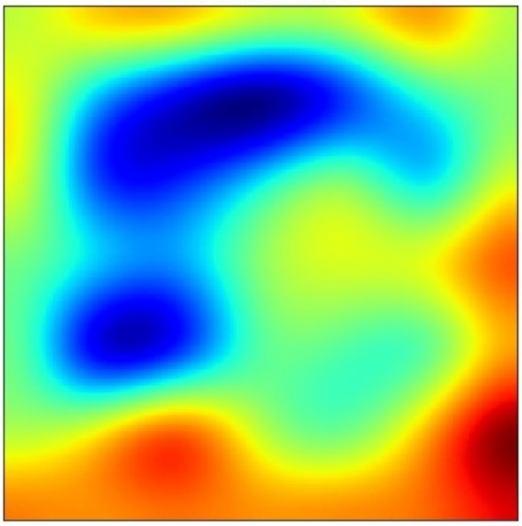
...al convolutional layer and weigh (multiply) every filter with the ...ure map. Grad-CAM involves the following steps:

1. Take the output feature map of the final convolutional layer. The shape of this feature map is 14x14x512 for VGG16
...n respect to the feature maps
...dients
...ding pooled gradients

...ding Class Activation Map below:

Now let's generate the Class activation map for the above image.

CLASS ACTIVATION MAP



Visualizing the Process – Layerwise Output Visualization

The starting layers of a CNN (https://courses.analyticsvidhya.com/courses/convolutional-neural-networks-cnn-from-scratch?utm_source=blog&utm_medium=understanding-visualizing-neural-networks) generally look for low-level features like edges. The features change as we go deeper into the model.

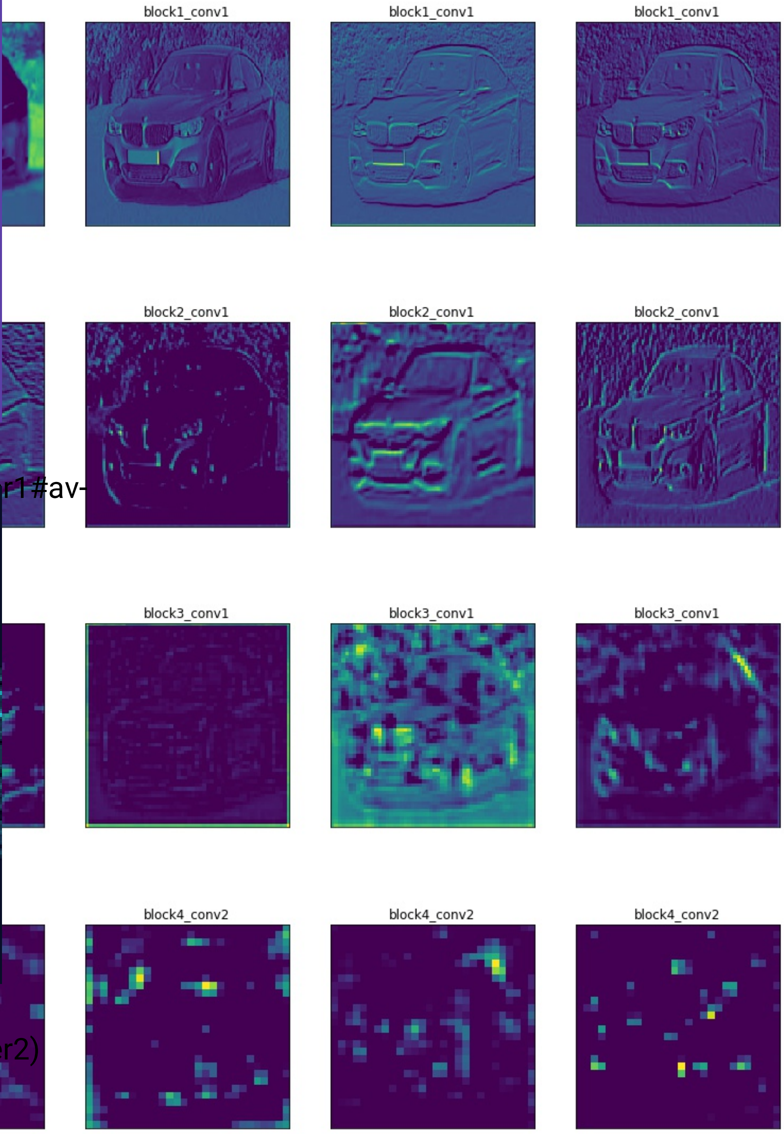
e model helps us see what features of the image are highlighted at the
rtant to fine-tune an architecture for our problems. Why? Because we
es and then decide which layers we want to use in our model.

p us compare the performance of different layers in the neural style
og/2018/12/guide-convolutional-neural-network-cnn/) problem.

nt layers of a VGG16 model:


```
ions
r will take the output
onv1','block3_conv1','block4_conv2']

], image.shape[1], image.shape[2]))
 outputs

puts=model.input,outputs=model.get_layer(layer_name).output)
layer_model.predict(image)


igsize=(20,20))
for i in range(4):
    for z in range(5):
        ax[i][z].imshow(outputs[i][0,:,:,z])
        ax[i][z].set_title(layer_names[i])
        ax[i][z].set_xticks([])
        ax[i][z].set_yticks([])
plt.savefig('layerwise_output.jpg')
```

st.github.com/saurabhpal97/9bf053b5f29052b2c8c60b69994ca123/raw/d15b56f716cc70f8a170a2b2b3c67111b586ff43/layerwise_output.py)
layerwise_output.py (https://gist.github.com/saurabhpal97/9bf053b5f29052b2c8c60b69994ca123#file-layerwise_output-py) hosted with ❤ by GitHub (https://github.com)



The above image shows the different features that are extracted from the image by every layer of VGG16 (except block 5). We can see that the starting layers correspond to low-level features like edges, whereas the later layers look at features like the roof, exhaust, etc. of the car.

End Notes

Visualization never ceases to amaze me. There are multiple ways to understand how a technique works, but visualizing it makes it a whole lot more fun. Here are a couple of resources you should check out:

- The process of feature extraction in neural networks is an active research area and has led to the development of awesome tools like Tensorspace and Activation Atlases (http://ai.googleblog.com/2019/03/exploring-neural-networks.html)

- TensorSpace (https://tensorspace.org/) is also a neural network visualization tool that supports multiple model formats. It lets you load your model and visualize it interactively. TensorSpace also has a playground where multiple architectures are available for visualization which you can play around with

dback on this article. I'll be happy to get into a discussion!

PP

=blog_article&utm_campaign=blog&pcampaignid=MKT-Other-global-all-co-prtnr-py-PartBadge-Mar2515-1)

 (https://apps.apple.com/us/app/analytics-

(https://ascendpro.analyticsvidhya.co

utm_source=blog&utm_medium=stickybanner1#av-

CSVIDHYA.COM/BLOG/TAG/COMPUTER-VISION/), DEEP LEARNING
DEEP-LEARNING/), NEURAL NETWORKS
NEURAL-NETWORKS/), PYTHON
PYTHON/)

(https://cbap.analyticsvidhya.com/?utm_source=blog&utm_medium=stickybanner2)

NEXT ARTICLE

Extracting and Analyzing 1000 Basketball Games using Pandas and Chartify

(https://www.analyticsvidhya.com/blog/2019/05/scraping-nba-data-analyze-1000-basketball-games-python/)

•••

PREVIOUS ARTICLE

Statistics for Data Science: Introduction to the Central Limit Theorem (with implementation in R)

(https://www.analyticsvidhya.com/blog/2019/05/statistics-101-introduction-central-limit-theorem/)

(https://www.analyticsvidhya.com/blog/author/saurabhpal97/)

Saurabh Pal (Https://Www.Analyticsvidhya.Com/Blog/Author/Saurabhpal97/)

A Data Science enthusiast and Software Engineer by training, Saurabh aims to work at the intersection of both fields.

al97/)

et a prompt response from the author. We request you to post this portal (https://discuss.analyticsvidhya.com/) to get your queries

JACQUE Reply

ticsvidhya.com/blog/2019/05/understanding-visualizing-neural-networks/#comment-

Reply

ticsvidhya.com/blog/2019/05/understanding-visualizing-neural-networks/#comment-

3D convolutional neural networks?

SAURABH PAL Reply

May 9, 2019 at 5:01 pm (https://www.analyticsvidhya.com/blog/2019/05/understanding-visualizing-neural-networks/#comment-158144)

Hi Xu, thanks. You can check out the following Github repo. https://github.com/OlesiaMidiana/3dcnn-vis (https://github.com/OlesiaMidiana/3dcnn-vis)

DIBIA Reply

May 8, 2019 at 6:32 am (https://www.analyticsvidhya.com/blog/2019/05/understanding-visualizing-neural-networks/#comment-158128)

Thank you for the exposure. What a wonderful piece of work!

SAURABH PAL Reply

May 9, 2019 at 5:08 pm (https://www.analyticsvidhya.com/blog/2019/05/understanding-visualizing-neural-networks/#comment-158146)

Hi, Dibia. Thanks.

VESA ALEXANDRU Reply

June 24, 2019 at 12:26 am (https://www.analyticsvidhya.com/blog/2019/05/understanding-visualizing-neural-

on (https://www.analyticsvidhya.com/blog/2021/06/donut-plots-data-visualization-with-python/)

ion in Python
021/06/lollipop-charts-advanced-data-visualization-in-python/)

ps://www.analyticsvidhya.com/blog/2021/06/support-vector-

ustering Techniques (Skill test Solution)
017/02/test-data-scientist-clustering/)

sics of Deep Learning (along with solution)
017/01/must-know-questions-deep-learning/)

hms (with Python and R Codes)
017/09/common-machine-learning-algorithms/)

30 Questions to test a data scientist on Tree Based Models (https://www.analyticsvidhya.com/blog/2017/09/30-questions-test-tree-based-models/)

25 Questions to test a Data Scientist on Support Vector Machines (https://www.analyticsvidhya.com/blog/2017/10/svm-skilltest/)

CAREER RESOURCES

16 Key Questions You Should Answer Before Transitioning into Data

Science (https://www.analyticsvidhya.com/16-key-questions-data-science-career-transition/?&utm_source=Blog&utm_medium=CareerResourceWidget)

VEMBER 23, 2020

at is A Business Analyst and What is the Role of a business analyst in ompany? (https://www.analyticsvidhya.com/blog/2021/06/what-is-a-siness-analyst-and-what-is-the-role-of-a-business-analyst-in-a-npany/?&utm_source=Blog&utm_medium=CareerResourceWidget)

NE 28, 2021

a Engineering – Concepts and Importance ps://www.analyticsvidhya.com/blog/2021/06/data-engineering-concepts-and-importance/?&utm_source=Blog&utm_medium=CareerResourceWidget)

NE 14, 2021

e's What You Need to Know to Become a Data Scientist! ps://www.analyticsvidhya.com/blog/2021/01/heres-what-you-need-know-to-become-a-data-scientist/?m_source=Blog&utm_medium=CareerResourceWidget)

IUARY 22, 2021

se 7 Signs Show you have Data Scientist Potential! ps://www.analyticsvidhya.com/blog/2020/12/these-7-signs-show-have-data-scientist-potential/?&utm_source=Blog&utm_medium=CareerResourceWidget)

DECEMBER 3, 2020

RECENT POSTS

Most powerful Python Functions apply() and lambda() (https://www.analyticsvidhya.com/blog/2021/07/most-powerful-python-functions-apply-and-lambda/)

JULY 16, 2021

What are the odds that you know about the odds? (https://www.analyticsvidhya.com/blog/2021/07/what-are-the-odds-and-odd-ratios-that-you-know-about-the-odds/)

JULY 16, 2021

Python most powerful functions: map(), filter(), and reduce() in 5 minutes (https://www.analyticsvidhya.com/blog/2021/07/python-most-powerful-functions-map-filter-and-reduce-in-5-minutes/)

JULY 16, 2021

Convolution Neural Network – Better Understanding! (https://www.analyticsvidhya.com/blog/2021/07/convolution-neural-network-better-understanding/)

JULY 16, 2021

(https://www.analyticsvidhya.com/)

Download App (https://play.google.com/store/apps/details?id=com.analyticsvidhya.android) (https://apps.apple.com/us/app/analytics-vidhya/id1470025572)

Analytics Vidhya

About Us (https://www.analyticsvidhya.com/about-me/)

Our Team (https://www.analyticsvidhya.com/about-me/team/)

Careers (https://www.analyticsvidhya.com/about-me/career-analytics-vidhya/)

Contact us (https://www.analyticsvidhya.com/contact/)

Data Science

Blog (https://www.analyticsvidhya.com/blog/)

Hackathon (https://datahack.analyticsvidhya.com/)

Discussions (https://discuss.analyticsvidhya.com/)

Apply Jobs (https://www.analyticsvidhya.com/jobs/)

Companies