Chapter 1: Setting Up OpenCV
Optional Features

- **Documentation**
  Installs the Python documentation file.

- **pip**
  Installs pip, which can download and install other Python packages.

- **tcl/tk and IDLE**
  Installs tkinter and the IDLE development environment.

- **Python test suite**
  Installs the standard library test suite.

- **py Launcher**
  For all users (requires elevation)
  Use Programs and Features to remove the 'py' launcher.
Advanced Options

- Install for all users
- Associate files with Python (requires the py launcher)
- Create shortcuts for installed applications
- Add Python to environment variables
- Precompile standard library
- Download debugging symbols
- Download debug binaries (requires VS 2015 or later)

Customize install location
C:\Python37
You will require write permissions for the selected location.

Back  Install  Cancel
Setup was successful

Special thanks to Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

New to Python? Start with the [online tutorial](#) and [documentation](#).

See [what's new](#) in this release.

**Disable path length limit**
Changes your machine configuration to allow programs, including Python, to bypass the 260 character "MAX_PATH" limitation.

```python
>>> import os
>>> os.getcwd()
'C:\Users\Alberto\python'
>>> Python 3.7.0 (v3.7.0:1bfc59b, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
>>> 'type help', 'copyright', 'credits' or 'license' for more information.
>>> '
```
Download PyCharm

Professional

Full-featured IDE for Python & Web development

Windows  meOS  Linux

Community

Lightweight IDE for Python & Scientific development

Download

System requirements
Installation Instructions
Previous versions

Free trial
Try Jupyter

You can try Jupyter out right now, without installing anything. Select an example below and you will get a temporary Jupyter server just for you, running on mybinder.org. If you like it, you can install Jupyter yourself.

- **Try Jupyter with Python**
  A tutorial introducing basic features of Jupyter notebooks and the IPython kernel.

- **Try JupyterLab**
  JupyterLab is the new interface for Jupyter notebooks and is ready for testing. Give it a try!

- **Try Jupyter with Julia**
  A basic example of using Jupyter with Julia.

- **Try Jupyter with R**
  A basic example of using Jupyter with R.

- **Try Jupyter with C++**
  A basic example of using Jupyter with C++.

- **Try Jupyter with Scheme**
  Explore the Guile Scheme programming language, featuring integration with Python.
Chapter 2: Image Basics in OpenCV
A Venn diagram illustrating the relationship between primary colors:

- **Green**
- **Cyan**
- **Yellow**
- **White**
- **Blue**
- **Magenta**
- **Red**
Red: 255
Green: 0
Blue: 255
Column 0  Column 1  Column m
Row 0
Row 1
Row n

pixel (y=n, x=1)
Getting and Setting methods in Python Using OpenCV

Introduction

This notebook is going to teach you the basic concepts you will need for accessing and manipulating pixels in images using OpenCV and Python (getting and setting methods) with BGR images. The test image, which is going to be used in this example, corresponds to the OpenCV logo image. To display an image in notebooks make sure the cell is in Markdown mode and use the following code: `![](filename.png)` - without the blank space before the imagename. This image is displayed next:

![OpenCV Logo](image.png)

So, let's start!

Load the image and see the properties of the loaded image

First of all, `import` the necessary packages:

```python
In [16]: #import required packages
import cv2
```
Chapter 3: Handling Files and Images

- Images
- Files

MP4, AVI

sys.argv
argparse
`cv2.VideoCapture()`
`cv2.imread()`

OpenCV and Python project

Project file inputs

Project file outputs

sys.argv[0] == 'sysargv_python.py'
sys.argv[1] == 'param 1'
sys.argv[2] == 'param 2'

param 1

sysargv_python.py

param 2
images

fourcc ('X', 'V', 'I', 'D')

width = 640

height = 480

fps = 30

video_demo.avi

cv2.VideoWriter()

video_demo.avi
capture.get(cv2.CAP_PROP_FOURCC)

828601953

00110001011000110111011001100001

00110001-01100011-01110110-01100001

97-118-99-49

a-v-c-1
Chapter 4: Constructing Basic Shapes in OpenCV

number of detected faces: ‘1’
face recognition: ‘lenna’
processing time: ‘0.26’ s
The image contains a table with keys and values mapping colors to their RGB values. The keys include 'blue', 'green', 'red', 'yellow', 'magenta', 'cyan', 'white', 'black', 'gray', 'rand', 'dark_gray', and 'light_gray'. The corresponding RGB values are given for each key.

Here is the Python code snippet for the color dictionary:

```python
colors = {
    'blue': (255, 0, 0),
    'green': (0, 255, 0),
    'red': (0, 0, 255),
    'yellow': (255, 255, 0),
    'magenta': (255, 0, 255),
    'cyan': (0, 255, 255),
    'white': (255, 255, 255),
    'black': (0, 0, 0),
    'gray': (125, 125, 125),
    'rand': np.random.randint(0, 255, size=(3)).tolist(),
    'dark_gray': (50, 50, 50),
    'light_gray': (255, 255, 255)
}
```

The code snippet demonstrates how to get one color from the dictionary by accessing `colors['light_gray']`. This would return `(220, 220, 220)`. The `rand` value is generated using `np.random.randint` and the `tolist()` method to convert the NumPy array into a Python list.
Figure 1

Dictionary with some predefined colors
Figure 1

LINE_4  LINE_AA  LINE_8  in cv2.line()
cv2.putText()

Mastering OpenCV4 with Python
Mastering OpenCV4 with Python
Mastering OpenCV4 with Python
cv2.putText() using all OpenCV fonts

FONT HERSHEY SIMPLEX
font hershey simplex

FONT HERSHEY PLAIN
font hershey plain

FONT HERSHEY DUPLEX
font hershey duplex

FONT HERSHEY COMPLEX
font hershey complex

FONT HERSHEY TRIPLEX
font hershey triplex

FONT HERSHEY COMPLEX SMALL
font hershey complex small

FONT HERSHEY SCRIPT SIMPLEX
font hershey script simplex

FONT HERSHEY SCRIPT COMPLEX
font hershey script complex
Double left click: add a circle
Simple right click: delete last circle
Double right click: delete all circle
Press 'q' to exit
Mastering OpenCV 4 with Python
The parametric equations of a translated circle with center \((x_0, y_0)\) and radius \(r\)

\[
x = x_0 + r \cos(t) \\
y = y_0 + r \sin(t)
\]

where, \(0 \leq t \leq 2\pi\)
Figure 1

very basic meme generator

Hello World

Goodbye World
Chapter 5: Image Processing Techniques

Splitting and merging channels in OpenCV

Smoothing techniques

original

cv2.filter2D() (5,5) kernel

cv2.filter2D() (10,10) kernel

cv2.blur()

cv2.boxFilter()

cv2.GaussianBlur()

cv2.medianBlur()

cv2.bilateralFilter() - small values

cv2.bilateralFilter() - big values
Sobel operator and cv2.addWeighted() to show the output

<table>
<thead>
<tr>
<th>Image</th>
<th>Gradient x</th>
<th>Gradient y</th>
<th>Sobel output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="gradient_x.png" alt="Gradient x" /></td>
<td><img src="gradient_y.png" alt="Gradient y" /></td>
<td><img src="sobel_output.png" alt="Sobel output" /></td>
</tr>
</tbody>
</table>

Bitwise operations (AND, OR, XOR, NOT)

<table>
<thead>
<tr>
<th>image 1</th>
<th>image 2</th>
<th>image 1 OR image 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="image 1" /></td>
<td><img src="image2.png" alt="image 2" /></td>
<td><img src="image1_or_image2.png" alt="image 1 OR image 2" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NOT (image 2)</th>
<th>NOT (image 1)</th>
<th>image 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="not_image2.png" alt="NOT (image 2)" /></td>
<td><img src="not_image1.png" alt="NOT (image 1)" /></td>
<td><img src="image3.png" alt="image 3" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>image 3 AND a loaded image</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image3_and_loaded.png" alt="image 3 AND a loaded image" /></td>
</tr>
</tbody>
</table>
Bitwise AND/OR between two images

AND operation

OR operation
Morpho operations - kernel_type='cv2.MORPH_RECT', kernel_size='(3,3)'

Color spaces in OpenCV
Custom color maps providing all values

```
gray
```

```
cv2.applyColorMap()
cv2.LUT()
cv2.applyColorMap()
cv2.LUT()
```

```
np.linspace(255, 180, 64)
```

```
((0, (255, 0, 255)),
 (0.25, (255, 0, 180)),
 (0.5, (255, 0, 120)),
 (0.75, (255, 0, 60)),
 (1.0, (255, 0, 0)))
```

```
np.linspace(0, 128, 64)
```

```
((0, (0, 255, 128)),
 (0.25, (128, 184, 64)),
 (0.5, (255, 128, 0)),
 (0.75, (64, 128, 224)),
 (1.0, (0, 128, 255)))
```
Custom color maps based on key colors and legend
Chapter 6: Constructing and Building Histograms

Figure 1

Grayscale histograms introduction

image with 50x50 regions of different tones of gray

image with 50x50 regions of different tones of gray
Figure 1

Grayscale masked histogram

gray

masked gray image

number of pixels

bins

number of pixels

bins
Custom visualization of histograms

gray

image

grayscale histogram (matplotlib)
color histogram (matplotlib)
grayscale histogram (custom)
color histogram (custom)
Figure 1

Comparing histogram (OpenCV, numpy, matplotlib)

Grayscale histogram (OpenCV): 3.50 ms  Grayscale histogram (Numpy): 12.00 ms  Grayscale histogram (Matplotlib): 583.42 ms

Grayscale histogram equalization with cv2.calcHist()

Grayscale histogram:

- Gray
- Grey lighter
- Grey darker

Grayscale equalized:

- Gray
grey lighter
grey darker
Figure 1: Color histogram equalization with cv2.calcHist() - not a good approach.
Color histogram equalization with cv2.calcHist() in the V channel
Figure 1

Histogram equalization using CLAHE

grey

gray CLAHE clipLimit=2.0

grey CLAHE clipLimit=5.0

grey CLAHE clipLimit=10.0

grey CLAHE clipLimit=20.0

color

cihlce on each channel (GRAY)

cihlce on L channel (LAB)

cihlce on V channel (HSV)

cihlce on Y channel (YUV)
Grayscale histogram equalization with cv2.calcHist() and CLAHE

gray  grayscale equalized  grayscale CLAHE
Grayscale histogram comparison

query img  img 1 CORREL 1.00000  img 2 CORREL 0.89083  img 3 CORREL 0.34628  img 4 CORREL 0.34569

query img  img 1 CHISQR 0.00000  img 2 CHISQR 0.34462  img 3 CHISQR 3.56062  img 4 CHISQR 4.05658

query img  img 1 INTERSECT 1.00000  img 2 INTERSECT 0.86386  img 3 INTERSECT 0.63674  img 4 INTERSECT 0.63674

query img  img 1 BHATTACHARYYA 0.00000  img 2 BHATTACHARYYA 0.20696  img 3 BHATTACHARYYA 0.40113  img 4 BHATTACHARYYA 0.40113
Chapter 7: Thresholding Techniques
Thresholding introduction

img with tones of gray - left to right: (0, 50, 100, 150, 200, 250)

threshold = 0

threshold = 50

threshold = 100

threshold = 150

threshold = 200

threshold = 250
Simple thresholding types

img tones of gray - left to right: (0,50,100,150,200,250)

THRESH_BINARY - thresh = 100 & maxValue = 255

THRESH_BINARY - thresh = 100 & maxValue = 220

THRESH_BINARY_INV - thresh = 100

THRESH_BINARY_INV - thresh = 100 & maxValue = 220

THRESH_TRUNC - thresh = 100

THRESH_TOZERO - thresh = 100

THRESH_TOZERO_INV - thresh = 100
Thresholding example

- Image
- Threshold = 60
- Threshold = 70
- Threshold = 80
- Threshold = 90
- Threshold = 100
- Threshold = 110
- Threshold = 120
- Threshold = 130
Otsu's binarization algorithm

image

gray img

Otsu's binarization
Otsu's binarization algorithm applying a Gaussian filter

image with noise

gray img with noise

Otsu's binarization (before applying a Gaussian filter)

Otsu's binarization (after applying a Gaussian filter)
Triangle binarization algorithm applying a Gaussian filter

image with noise

gray img with noise

Triangle binarization (before applying a Gaussian filter)

Triangle binarization (after applying a Gaussian filter)
Thresholding scikit-image (Otsu’s binarization example)

image

gray img

Otsu’s binarization (scikit-image)
Thresholding scikit-image (Otsu, Triangle, Niblack, Sauvola)

- Image
- Grey Image
- Otsu's Binarization (scikit-image)

- Triangle Binarization (scikit-image)
- Niblack's Binarization (scikit-image)
- Sauvola's Binarization (scikit-image)
Chapter 8: Contour Detection, Filtering, and Drawing
Hu moment invariants properties

original  rotation  reflection
Functionality related to contours

- Image and extreme points
- cv2.boundingRect()
- cv2.minAreaRect()
- cv2.minEnclosingCircle()
- cv2.ellipse()
- cv2.approxPolyDP()
Sort contours by size

image

result

[83]
Shape recognition based on cv2.approxPolyDP()
Matching contours (against a perfect circle) using cv2.matchShapes()
Chapter 9: Augmented Reality
ORB descriptors and Brute-Force (BF) matcher

matches between the two images
Feature matching & homography computation

feature matching
Aruco markers creation

marker_DICT_7X7_250_600_1   marker_DICT_7X7_250_600_2   marker_DICT_7X7_250_600_3
QR code detection
rectified QR code
decoded data: https://github.com/PacktPublishing/Mastering-OpenCV-4-with-Python
Chapter 10: Machine Learning with OpenCV
K-means clustering algorithm

data to be clustered
K-means clustering algorithm

data

clustered data and centroids (K = 4)

Color quantization using K-means clustering algorithm

original image

color quantization (k = 3)

color quantization (k = 5)

color quantization (k = 10)

color quantization (k = 20)

color quantization (k = 40)
Color quantization using K-means clustering algorithm

- Original image
- Color quantization (k = 3)
- Color quantization (k = 5)
- Color quantization (k = 10)
- Color quantization (k = 20)
- Color quantization (k = 40)

k = 3
k = 5
k-NN algorithm: sample green point is classified as red (k = 3)
k-NN handwritten digits recognition

Accuracy of the K-NN model varying k
**k-NN handwritten digits recognition**

Accuracy of the KNN model varying both k and the percentage of images to train/test

---

![Graph showing accuracy of KNN model varying k and percentage of images for training/test.](image)

---

![Images of handwritten digits: 01579 01579](image)
k-NN handwritten digits recognition

Accuracy of the k-NN model varying both k and the percentage of images to train/test with pre-processing.
SVM introduction
Visual representation of SVM model
Chapter 11: Face Detection, Tracking, and Recognition
Face detection using haar feature-based cascade classifiers

detectMultiScale(frontalface_alt2): 2

detectMultiScale(frontalface_default): 4

getFacesHAAR(frontalface_alt2): 3

getFacesHAAR(frontalface_default): 8
Cat face detection using haar feature-based cascade classifiers

\[ \text{detectMultiScale(frontalcatface)}: 2 \]

\[ \text{detectMultiScale(frontalcatface\_extended)}: 3 \]

\[ \text{getFacesHAAR(frontalcatface)}: 3 \]

\[ \text{getFacesHAAR(frontalcatface\_extended)}: 3 \]
Face detection using OpenCV DNN face detector

DNN face detector: 3

99.922%  97.495%  79.345%
**Face detection using dlib frontal face detector**

<table>
<thead>
<tr>
<th>detector(gray, 0): 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Face detection" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>detector(gray, 1): 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image2" alt="Face detection" /></td>
</tr>
</tbody>
</table>
Face detection using dlib CNN face detector

cnn_face_detector(img, 0): 3
Facial landmarks detection using face_recognition

68 facial landmarks

5 facial landmarks
Use '1' to re-initialize tracking
detecting a face to initialize tracking...
Use '1' to re-initialize tracking
tracking the face
Use left click of the mouse to select the object to track
Use '1' to start tracking, '2' to reset tracking and 'q' to exit
not tracking
Use left click of the mouse to select the object to track.
Use '1' to start tracking, '2' to reset tracking and 'q' to exit tracking.
Chapter 12: Introduction to Deep Learning
<table>
<thead>
<tr>
<th>Model</th>
<th>PASCAL VOC 2007 (%)</th>
<th>PASCAL VOC 2010 (%)</th>
<th>PASCAL VOC 2012 (%)</th>
<th>COCO 2015 (IoU=0.5) (%)</th>
<th>COCO 2015 (IoU=0.75) (%)</th>
<th>COCO 2016 (Official Metric) (%)</th>
<th>COCO 2016 (IoU=0.5) (%)</th>
<th>COCO 2016 (IoU=0.75) (%)</th>
<th>COCO 2016 (Official Metric) (%)</th>
<th>Real Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-CNN (2014)</td>
<td>-</td>
<td>62.4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>No</td>
</tr>
<tr>
<td>Fast R-CNN (2015)</td>
<td>70.0</td>
<td>68.8</td>
<td>68.4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>No</td>
</tr>
<tr>
<td>Faster R-CNN (2015)</td>
<td>78.8</td>
<td>-</td>
<td>75.9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>No</td>
</tr>
<tr>
<td>R-FCN (2016)</td>
<td>82.0</td>
<td>-</td>
<td>-</td>
<td>53.2</td>
<td>-</td>
<td>31.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>No</td>
</tr>
<tr>
<td>YOLO (2016)</td>
<td>63.7</td>
<td>57.9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>SDD (2016)</td>
<td>83.2</td>
<td>-</td>
<td>82.2</td>
<td>48.5</td>
<td>30.3</td>
<td>31.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>No</td>
</tr>
<tr>
<td>YOLO V2 (2016)</td>
<td>78.6</td>
<td>-</td>
<td>-</td>
<td>44.0</td>
<td>19.2</td>
<td>21.6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>NASNet (2016)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>43.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>No</td>
</tr>
<tr>
<td>Mask R-CNN (2017)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>62.3</td>
<td>43.3</td>
<td>39.8</td>
<td>-</td>
<td>No</td>
</tr>
</tbody>
</table>

**cv2.dnn.blobFromImage() visualization**

- img from blob (300, 300, 3)
- img from blob swap (300, 300, 3)
- img from blob mean (300, 300, 3)
- img from blob mean swap (300, 300, 3)
cv2.dnn.blobFromImages() visualization with cropping

img 1 from blob (300, 300, 3)

img 1 from blob cropped (300, 300, 3)

img 2 from blob (300, 300, 3)

img 2 from blob cropped (300, 300, 3)
OpenCV DNN face detector when feeding several images

input img 1

output img 1

input img 2

output img 2
OpenCV DNN face detector when feeding several images and cropping

input img 1

output cropped img 1

input img 2

output cropped img 2
Image classification with OpenCV using AlexNet and caffe pre-trained models

AlexNet and caffe pre-trained models

label: church probability: 83.26%
Image classification with OpenCV using GoogLeNet and caffe pre-trained models

GoogLeNet and caffe pre-trained models

label: church probability: 90.83%
Image classification with OpenCV using ResNet-50 and caffe pre-trained models

ResNet-50 and caffe pre-trained models

*label: church probability: 99.55%*

Image classification with OpenCV using SqueezeNet (v1.1) and caffe pre-trained models

SqueezeNet (v1.1) and caffe pre-trained models

*label: church probability: 99.55%*
Object detection using OpenCV DNN module and MobileNet-SSD

MobileNet-SSD for object detection

aeroplane: 0.9859072
bicycle: 0.9675555
bird: 0.9943400
sheep: 0.97885233
horse: 0.8097682
boat: 0.5663269
pottedplant: 0.325218
bus: 0.9995572
car: 0.9977561
chair: 0.9940455
Object detection using OpenCV DNN module and YOLO V3

YOLO V3 for object detection

TensorBoard

my_multiplication
Operation: Mul
Attributes (1)
- T
Inputs (2)
- X_1
- X_2
Outputs (0)
Linear regression using TensorFlow

Training Data

Linear Regression Result

Original data

Fitted line
Linear regression using TensorFlow

Training Data

Original data

Linear Regression Result

Original data
Fitted line

Predicting new points

Original data
Fitted line
New predicted data
<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
</tbody>
</table>
Linear regression using Keras

Training Data

Linear Regression Result

Original data

Fitted line

Predicting new points

Original data

Fitted line

New predicted data
Chapter 13: Mobile and Web Computer Vision with Python and OpenCV

Python web frameworks

Full-stack frameworks

Non full-stack frameworks

django

Flask
web development, one drop at a time
Hello World!

Hello World!
User: Hello World!
Using face API

face detection
Using cat detection API

cat detection
Keras Applications: Models for image classification with weights trained on ImageNet

<table>
<thead>
<tr>
<th>Model</th>
<th>Size</th>
<th>Top-1 Accuracy</th>
<th>Top-5 Accuracy</th>
<th>Parameters</th>
<th>Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xception</td>
<td>88 MB</td>
<td>0.790</td>
<td>0.945</td>
<td>22,910,480</td>
<td>126</td>
</tr>
<tr>
<td>VGG16</td>
<td>528 MB</td>
<td>0.713</td>
<td>0.901</td>
<td>138,357,544</td>
<td>23</td>
</tr>
<tr>
<td>VGG19</td>
<td>549 MB</td>
<td>0.713</td>
<td>0.900</td>
<td>143,667,240</td>
<td>26</td>
</tr>
<tr>
<td>ResNet50</td>
<td>99 MB</td>
<td>0.749</td>
<td>0.921</td>
<td>25,636,712</td>
<td>168</td>
</tr>
<tr>
<td>InceptionV3</td>
<td>92 MB</td>
<td>0.779</td>
<td>0.937</td>
<td>23,851,784</td>
<td>159</td>
</tr>
<tr>
<td>InceptionResNetV2</td>
<td>215 MB</td>
<td>0.803</td>
<td>0.953</td>
<td>55,873,736</td>
<td>572</td>
</tr>
<tr>
<td>MobileNet</td>
<td>16 MB</td>
<td>0.704</td>
<td>0.895</td>
<td>4,253,864</td>
<td>88</td>
</tr>
<tr>
<td>MobileNetV2</td>
<td>14 MB</td>
<td>0.713</td>
<td>0.901</td>
<td>3,538,984</td>
<td>88</td>
</tr>
<tr>
<td>DenseNet121</td>
<td>33 MB</td>
<td>0.750</td>
<td>0.923</td>
<td>8,062,504</td>
<td>121</td>
</tr>
<tr>
<td>DenseNet169</td>
<td>57 MB</td>
<td>0.762</td>
<td>0.932</td>
<td>14,307,880</td>
<td>169</td>
</tr>
<tr>
<td>DenseNet201</td>
<td>80 MB</td>
<td>0.773</td>
<td>0.936</td>
<td>20,242,984</td>
<td>201</td>
</tr>
<tr>
<td>NASNetMobile</td>
<td>23 MB</td>
<td>0.744</td>
<td>0.919</td>
<td>5,326,716</td>
<td>-</td>
</tr>
<tr>
<td>NASNetLarge</td>
<td>343 MB</td>
<td>0.825</td>
<td>0.960</td>
<td>88,949,818</td>
<td>-</td>
</tr>
</tbody>
</table>
Image classification in Keras using several pre-trained models

source image

classification results

InceptionV3: sports_car, 0.53
VGG16: minivan, 0.38
VGG19: minivan, 0.23
ResNet50: sports_car, 0.29
MobileNet: sports_car, 0.29
Xception: sports_car, 0.34
NASNetMobile: sports_car, 0.55
DenseNet121: sports_car, 0.65
Image classification in Keras using several pre-trained models

Source image

Classification results

InceptionV3: tabby, 0.39
VGG16: tabby, 0.22
VGG19: Cardigan, 0.16
ResNet50: doormat, 0.29
MobileNet: tabby, 0.18
Xception: Egyptian_cat, 0.24
NASNetMobile: tabby, 0.39
DenseNet121: spotlight, 0.43
Using Keras Deep Learning REST API

Classification results (NASNetMobile)

1. sports_car: 0.5461
2. convertible: 0.2798
3. grille: 0.0373
4. car_wheel: 0.0268
5. beach_wagon: 0.0142
Create new web app

Your web app's domain name

Your account doesn't support custom domain names, so your PythonAnywhere web app will live at opencv.pythonanywhere.com.

Want to change that? Upgrade now!

Otherwise, just click "Next" to continue.
Create new web app

Quickstart new Flask project

Enter a path for a Python file you wish to use to hold your Flask app. If this file already exists, its contents will be overwritten with the new app.

Path

/home/opencv/mysite/flask_app.py

Code:
What your site is running.

Source code: /home/opencv/mysite
Working directory: /home/opencv/
WSGI configuration file: /var/www/opencv_pythonanywhere_com_wsgi.py
Python version: 3.6

[ 154 ]
Using face API at http://opencv.pythonanywhere.com/detect

face detection