pyimagesearch
be awesome at building image search engines

# Convert URL to image with Python and OpenCV

by **Adrian Rosebrock** on March 2, 2015 in **Tutorials**


Today's blog post comes directly from my own personal repository of utility functions.

Over the past month I've gotten a handful of PyImageSearch readers emailing in and ask
then convert it to OpenCV format (without writing it to disk and then reading it back) — ar

And as a bonus we'll also see how we can utilize scikit-image to download an image from
trip you up along the way.

Continue reading to learn how to convert a URL to an image using Python and OpenCV.

**Looking for the source code to this post?**
**Jump right to the downloads section.**

**OpenCV and Python versions:**
In order to run this example, you'll need **Python 2.7** and **OpenCV 2.4.X**.

# Method #1: OpenCV, NumPy, and urllib

The first method we'll explore is converting a URL to an image using the OpenCV, NumPy, and the urllib libraries. Open up a new file, name it `url_to_image.py` , and let's get started:

Convert URL to image with Python and OpenCV — Python

```
1  # import the necessary packages
2  import numpy as np
3  import urllib
4  import cv2
5
6  # METHOD #1: OpenCV, NumPy, and urllib
7  def url_to_image(url):
8      # download the image, convert it to a NumPy array, and then read
9      # it into OpenCV format
10     resp = urllib.urlopen(url)
11     image = np.asarray(bytearray(resp.read()), dtype="uint8")
12     image = cv2.imdecode(image, cv2.IMREAD_COLOR)
13
14     # return the image
15     return image
```

The first thing we'll do is import our necessary packages. We'll use NumPy for converting the byte-sequence from the download to a NumPy array, `urllib` to perform the actual request, and `cv2` for our OpenCV bindings.

We then define our `url_to_image` function on **Line 7**. This function requires a single argument, `url` , which is the URL of the image we want to download.

Next, we utilize the `urllib` library to open a connection to the supplied URL on **Line 10**. The raw byte-sequence from the request is then converted to a NumPy array on **Line 11**.

At this point the NumPy array is a 1-dimensional array (i.e. a long list of pixels). To reshape the array into a 2D format, assuming 3 components per pixel (i.e. the Red, Green, and Blue components, respectively), we make a call to `cv2.imdecode` on **Line 12**. Finally, we return the decoded image to the calling function on **Line 15**.

Alright, time to put this function to work:

Convert URL to image with Python and OpenCV — Python

```
17 # initialize the list of image URLs to download
18 urls = [
19     "https://www.pyimagesearch.com/wp-content/uploads/2015/01/opencv_logo.png",
20     "https://www.pyimagesearch.com/wp-content/uploads/2015/01/google_logo.png",
21     "https://www.pyimagesearch.com/wp-content/uploads/2014/12/adrian_face_detection_sidebar.png",
22 ]
23
24 # loop over the image URLs
25 for url in urls:
26     # download the image URL and display it
27     print "downloading %s" % (url)
28     image = url_to_image(url)
29     cv2.imshow("Image", image)
30     cv2.waitKey(0)
```

**Lines 18-21** define a list of image URLs that we are going to download and convert to OpenCV format.

We start looping over each of these URLs on **Line 25**, make a call to our `url_to_image` function on **Line 28**, and then finally display our downloaded image to our screen on **Lines 29 and 30**. At this point our image can be manipulated with any other OpenCV functions as we normally would.

To see our work in action, open up a terminal and execute the following command:

Convert URL to image with Python and OpenCV — Shell

```
1 $ python url_to_image.py
```

If all goes well, you should first see the OpenCV logo:

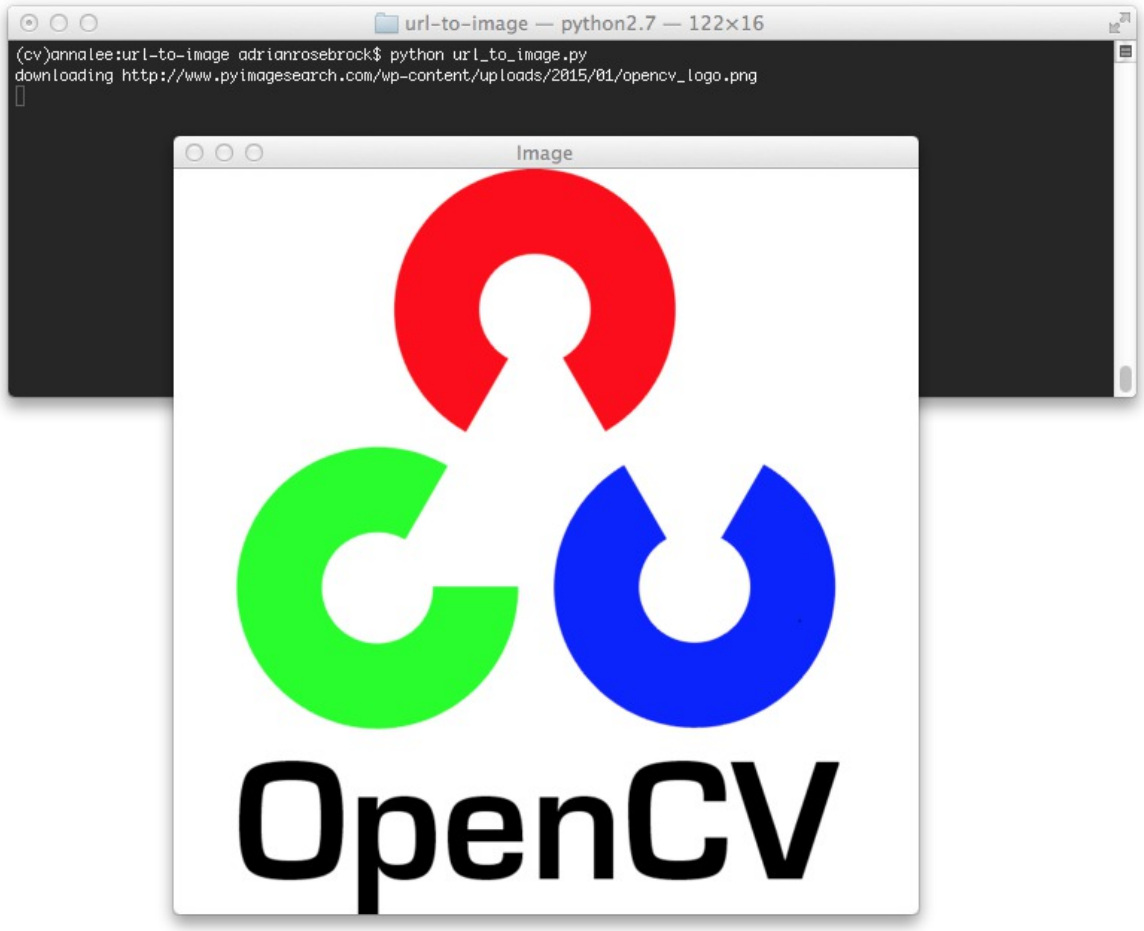



**Figure 1:** Downloading the OpenCV logo from a URL and converting it to OpenCV format.

And next the Google logo:

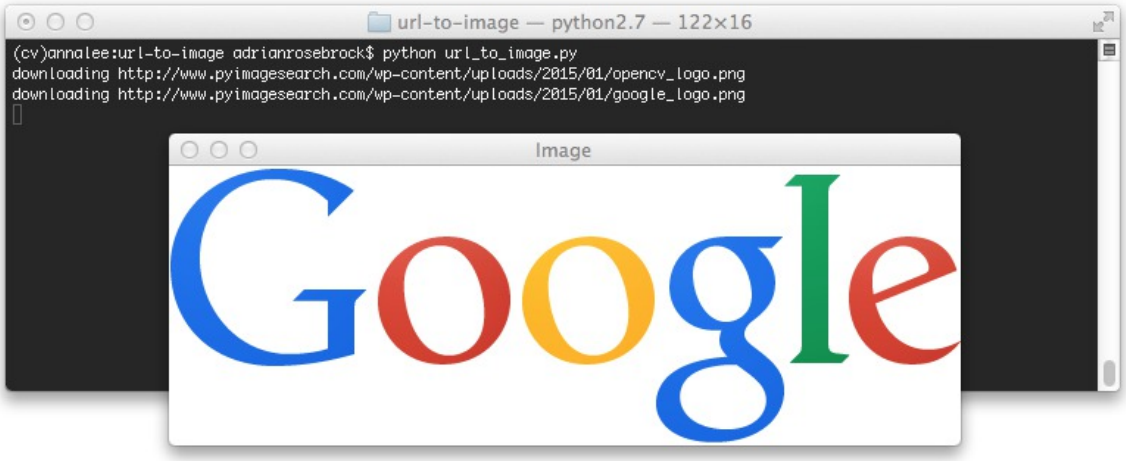



**Figure 2:** Downloading the Google logo from a URL and converting it to OpenCV format.

And here's an example of me demonstrating face detection in my book, *Practical Python and OpenCV*:

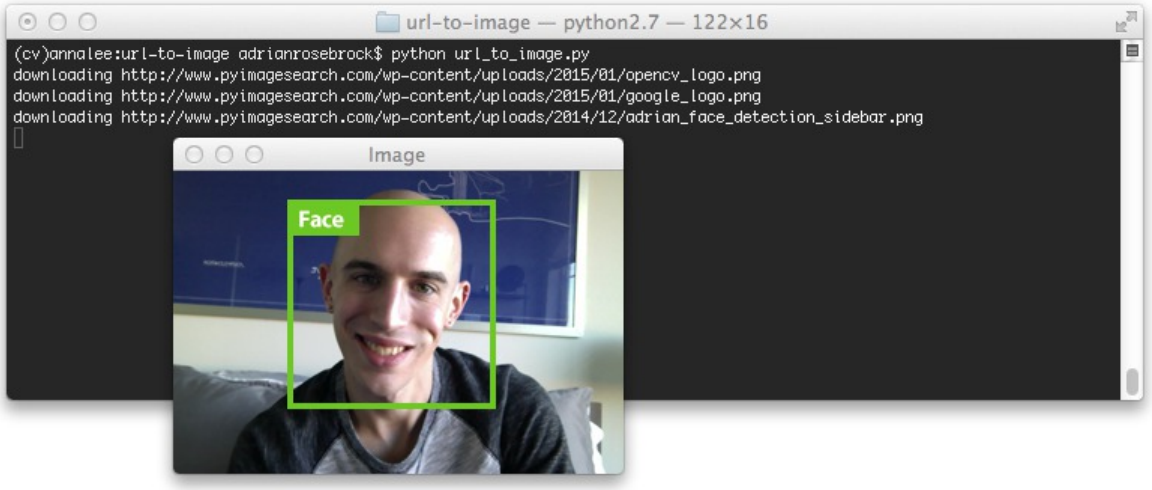



**Figure 3:** Converting an image URL to OpenCV format with Python.

Now, let's move on to the alternative method to downloading an image and converting it to OpenCV format.

# Method #2: scikit-image

The second method assumes that you have the scikit-image library installed on your system. Let's take a look at how we can leverage scikit-image to download an image from a URL and convert it to OpenCV format:

Convert URL to image with Python and OpenCV — Python

```
32 # METHOD #2: scikit-image
33 from skimage import io
34
35 # loop over the image URLs
36 for url in urls:
37     # download the image using scikit-image
38     print "downloading %s" % (url)
39     image = io.imread(url)
40     cv2.imshow("Incorrect", image)
41     cv2.imshow("Correct", cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
42     cv2.waitKey(0)
```

One of the nice aspects of the scikit-image library is that the `imread` function in the `io` sub-package can tell the difference between a path to an image on disk and a URL (**Line 39**).

***However, there is an important gotcha that can really trip you up!***

OpenCV represents images in BGR order — whereas scikit-image represents images in RGB order. If you use the scikit-image `imread` function and want to utilize OpenCV functions after downloading the image, you need to take special care to convert the image from RGB to BGR (**Line 41**).

If you don't take this extra step, you may obtain incorrect results:

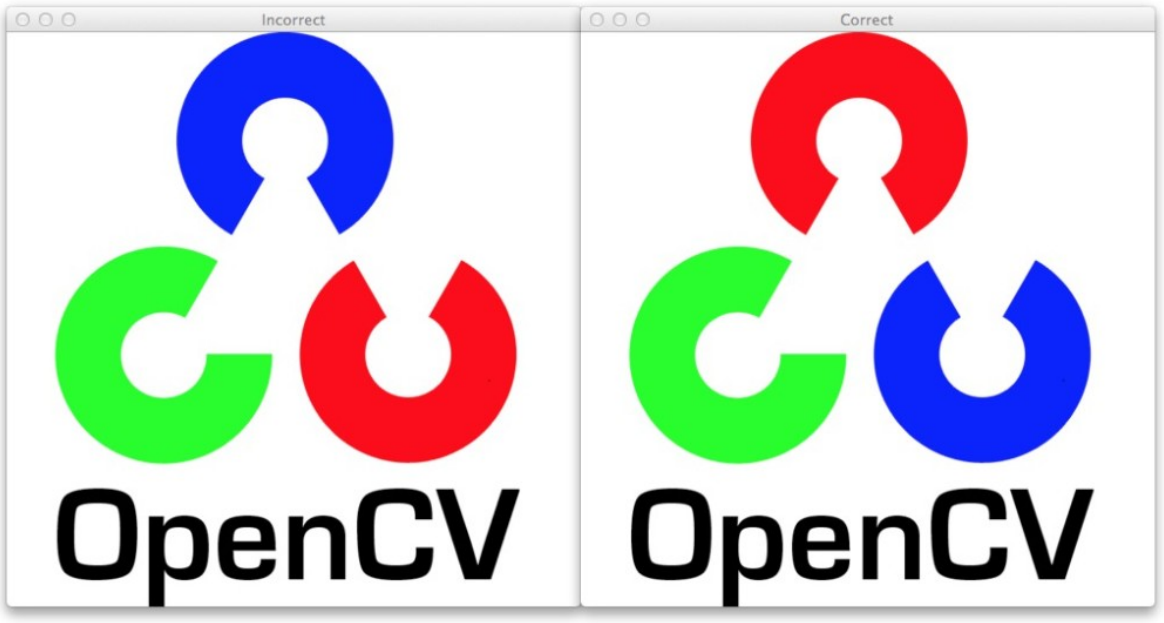



**Figure 4:** Special care needs to be taken to convert from RGB to BGR when using scikit-image to convert a URL to an image. The image on the *left* is incorrectly specified in the RGB order. The image on the *right* correctly displays the image after it is converted from RGB to BGR order.

Take a look at the Google logo below to make this point even more clear:


**Figure 5:** Order matters. Be sure to convert from RGB to BGR order or you might be tracking down a hard-to-find bug.

So there you have it! Two methods to convert a URL to an image using Python, OpenCV, urllib, and scikit-image.

# Summary

In this blog post we learned about two methods to download an image from a URL and convert it to OpenCV format using Python and OpenCV.

The first method is to use the `urllib` Python package to download the image, convert it to an array using NumPy, and finally reshape the array using OpenCV to construct our image.

The second method is to use the `io.imread` function of scikit-image.

So which method is better?

It all depends on your setup.

If you already have scikit-image installed, I would use the `io.imread` function (just don't forget to convert from RGB to BGR if you are using OpenCV functions). And if you do not have scikit-image installed, I would hand-roll the `url_to_image` function detailed at the beginning of this article.

I'll also be adding this function to the imutils package on GitHub soon.

## Downloads:

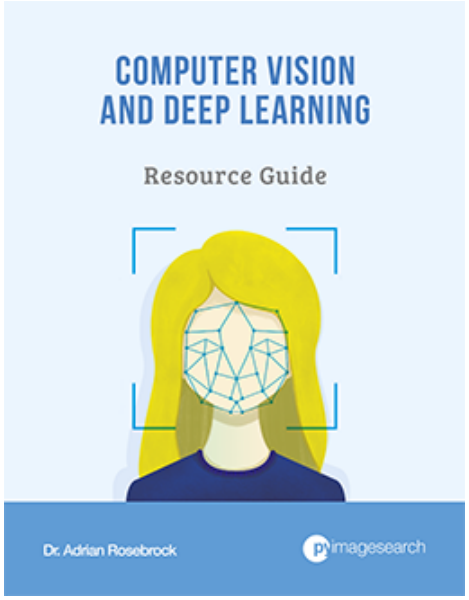
If you would like to download the code and images used in this post, please enter your email address in the form below. Not only will you get a .zip of the code, I'll also send you a **FREE 17-page Resource Guide on Computer Vision, OpenCV, and Deep Learning.** Inside you'll find my hand-picked tutorials, books, courses, and libraries to help you master CV and DL! Sound good? If so, enter your email address and I'll send you the code immediately!

**Email address:**

Your email address

DOWNLOAD THE CODE!

### Resource Guide (it's totally free).




Enter your email address below to get my **free 17-page Computer Vision, OpenCV, and Deep Learning Resource Guide PDF**. Inside you'll find my hand-picked tutorials, books, courses, and Python libraries to help you master computer vision and deep learning!

Your email address

DOWNLOAD THE GUIDE!

**download image**, **image processing**, **url to image**

‹ Install OpenCV and Python on your Raspberry Pi 2 and B+

Capturing mouse click events with Python and OpenCV ›

## 21 Responses to *Convert URL to image with Python and OpenCV*

**Captain DeadBones** March 13, 2015 at 5:40 pm #

REPLY

Great post! Awesome idea.

**Adrian Rosebrock** March 13, 2015 at 6:23 pm #

REPLY

I'm glad you enjoyed it! 🙂

**newbie** September 5, 2015 at 8:20 am #

REPLY

Hi,

Line 41 shouldn't be cv2.COLOR_RGB2BGR instead of cv2.COLOR_BGR2RGB

**Adrian Rosebrock** September 5, 2015 at 1:30 pm #

REPLY

The scikit-image library represents images in RGB order, whereas OpenCV represents images in BGR order. So when you download the image via scikit-image's `io.imread` function, your image is in RGB order. Thus, you need to reverse it. This can be done using raw NumPy array functions, or you can (somewhat confusingly) use `cv2.COLOR_BGR2RGB` to flip the order of the channels. Remember, an image is just a NumPy array and it has no notion or understanding of what color space it is in.

**Anon** October 29, 2015 at 11:23 am #

REPLY

What if there is a password? Like a camera needing simple user/pass authentication.

Thanks!

**Adrian Rosebrock** November 3, 2015 at 10:36 am #

REPLY

That will make things a little more complicated. I would suggest looking into the requests Python library which supports simple authentication.

**Inês Martins** October 30, 2015 at 10:16 am #

REPLY

I tried method #2 and I am getting this error:

urllib2.HTTPError: HTTP Error 403: Forbidden

**Adrian Rosebrock** November 3, 2015 at 10:28 am #

REPLY

If you are getting an error related to `urllib2` then the image URL you are requesting is not valid or can not be found. A 403 error is a common error for a server to return.

**Luis** October 7, 2016 at 12:32 pm #

REPLY

What if the image is not available? how could i just ignore such image and continue with next?

**Adrian Rosebrock** October 11, 2016 at 1:16 pm #

REPLY

If the image cannot be downloaded you can detect this and catch the exception and move on to the next URL.

**udit** December 4, 2017 at 5:08 am #

REPLY

You can look at the status code for the call using url.response.code, if its value is 200, you are good else move on to the next image.

**judson antu** January 26, 2017 at 5:20 am #

REPLY

can we do it the opposite way? posting an image as url in a server?

**Adrian Rosebrock** January 26, 2017 at 8:14 am #

REPLY

Hi Judson — can you elaborate on what you mean by “posting an image as URL in a server”? I’m not sure what you mean.

**judson antu** January 26, 2017 at 11:49 pm #

REPLY

im, sorry if i have confused you. here you have displayed the opencv logo from http/pyimage search…….opencv.png which is a url right? can we post an image in my rpi as a url like this, so that some body else can download this image in their own system?

**Adrian Rosebrock** January 28, 2017 at 6:57 am #

REPLY

If you want to take an image and upload it to a server, I would suggest referring to this tutorial.

**Kyle Hounslow** May 8, 2017 at 1:02 pm #

REPLY

My man! Always got what I need.
I used this to receive and decode images for one of my flask APIs, thank you.
I’ll reply to this comment with some demo code once finished.

**Kyle Hounslow** May 16, 2017 at 1:27 pm #

REPLY

See this gist: https://gist.github.com/kylehounslow/767fb72fde2ebdd010a0bf4242371594

**Adrian Rosebrock** May 17, 2017 at 9:54 am #

REPLY

Thanks for sharing Kyle!

**Tiffany** May 4, 2018 at 8:22 pm #

REPLY

Has to add a bit for it to work in Python 3.x.

```
import urllib.request
.
.
.
resp = urllib.request.urlopen(url)
```

**Thomas** August 8, 2018 at 6:13 pm #

REPLY

The url_to_image.py method isn't working for me. At all. I've tried every way to do this. It isn't working.

**Adrian Rosebrock** August 9, 2018 at 2:48 pm #

REPLY

Hey Thomas — could you be a bit more specific in what you mean by "isn't working"? Are you getting an error? Are the results not what you expect? Keep in mind I can't keep if you aren't more detailed.

## Leave a Reply

Name (required)

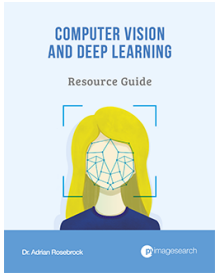
Email (will not be published) (required)

Website

SUBMIT COMMENT

Search...

**Resource Guide (it's totally free).**




Get your **FREE 17 page Computer Vision, OpenCV, and Deep Learning Resource Guide PDF.** Inside you'll find my hand-picked tutorials, books, courses, and libraries to help you master CV and DL.

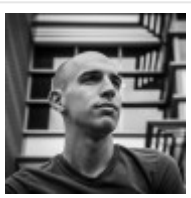
Download for Free!

**Deep Learning for Computer Vision with Python Book — OUT NOW!**


You're interested in deep learning and computer vision, *but you don't know how to get started.* Let me help. **My new book will teach you all you need to know about deep learning.**

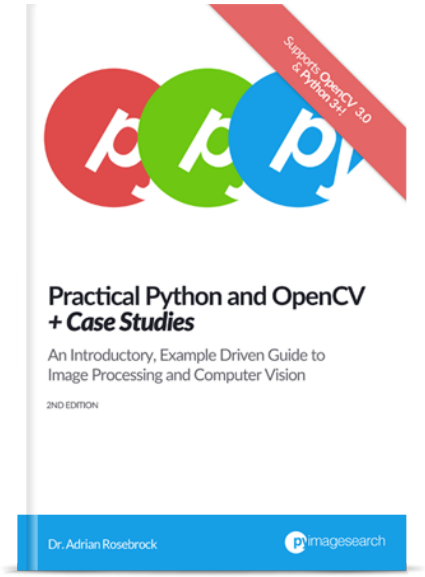
CLICK HERE TO MASTER DEEP LEARNING

**You can detect faces in images & video.**



I'm an entrepreneur and Ph.D who has launched two successful image search engines, ID My Pill and Chic Engine. I'm here to share my tips, tricks, and hacks I've learned along the way.

### Learn computer vision in a single weekend.




Want to learn computer vision & OpenCV? I can teach you in a **single weekend**. I know. It sounds crazy, but it's no joke. My new book is your **guaranteed, quick-start guide** to becoming an OpenCV Ninja. So why not give it a try? Click here to become a computer vision ninja.

CLICK HERE TO BECOME AN OPENCV NINJA

### Subscribe via RSS

**Never miss a post!** Subscribe to the PyImageSearch RSS Feed and keep up to date with my image search engine tutorials, tips, and tricks

POPULAR

**Install guide: Raspberry Pi 3 + Raspbian Jessie + OpenCV 3**
APRIL 18, 2016

**Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi**
SEPTEMBER 4, 2017

**Install OpenCV and Python on your Raspberry Pi 2 and B+**
FEBRUARY 23, 2015

**Home surveillance and motion detection with the Raspberry Pi, Python, OpenCV, and Dropbox**
JUNE 1, 2015

**Ubuntu 16.04: How to install OpenCV**
OCTOBER 24, 2016

**How to install OpenCV 3 on Raspbian Jessie**
OCTOBER 26, 2015

**Basic motion detection and tracking with Python and OpenCV**
MAY 25, 2015

Find me on **Twitter**, **Facebook**, **Google+**, and **LinkedIn**.