

<https://www.pyimagesearch.com/2019/09/16/install-opencv-4-on-raspberry-pi-4-and-raspbian-buster/>

# Install OpenCV 4 on Raspberry Pi 4 and Raspbian Buster

by [Adrian Rosebrock](#) on September 16, 2019 in [Embedded](#), [IoT](#), [OpenCV](#), [Raspberry Pi](#), [Tutorials](#)

In this tutorial you will learn how to install OpenCV 4 on the Raspberry Pi 4 and Raspbian Buster.

You will learn how to install OpenCV 4 on Raspbian Buster via both:

1. **A simple pip-install method** (which can be completed in a matter of minutes)
2. **Compiling from source** (which will take longer but will give you access to the full, optimized install of OpenCV)

To learn more about installing OpenCV 4 on the Raspberry Pi 4 and Raspbian Buster, *just keep reading*.

## Install OpenCV 4 on Raspberry Pi 4 and Raspbian Buster

In this tutorial we will install and test OpenCV 4 on Raspbian Buster in five simple, easy-to-follow steps.

If you' ve ever compiled OpenCV from scratch before, you know that the process is *especially time-consuming* and even *painstakingly frustrating* if you miss a key step or if you are new to Linux and Bash.

In Q4 2018, a [new, faster method for installing OpenCV on the Raspberry Pi](#) (i.e., a pip install) was made possible thanks to the hard work of the following people:

- [Olli-Pekka Heinisuo](#) — maintainer of the [opencv-contrib-python package on PyPi](#)

- [Ben Nuttall](#) — from the Raspberry Pi community-run [piwheels.org](#), a Python package repository providing ARM wheels (i.e., pre-compiled binaries packages) for the Raspberry Pi
- [Dave Jones](#) — creator of the [picamera](#) Python module

Installing OpenCV via pip is easier than ever. In fact, you can be up and running (**Step #1 – Step #4a**) in less than 10 minutes.

### **But what' s the catch?**

Using pip to install OpenCV is great, but for some projects (including many educational projects on PyImageSearch.com and in my books/courses) **you might want the complete install of OpenCV (which the pip install *won' t* give you).**

Don' t worry, I' ve got you covered in **Step #4b** below — you' ll learn to use CMake and Make to compile OpenCV 4 on BusterOS from scratch.

Let' s dive in!

## **Before we begin: Grab your Raspberry Pi and flash BusterOS to your microSD**

Let' s review the **hardware requirements** for this tutorial:

- **Raspberry Pi:** Despite the title of this tutorial, you may use the **Raspberry Pi hardware including 3B, 3B+, or 4B** to install OpenCV 4. These instructions ***only apply to Raspbian Buster***, however.
- **32GB microSD:** I recommend the high-quality SanDisk 32GB 98Mb/s cards. Here' s an example on [Amazon](#) (however you can purchase them on your favorite online distributor).
- **microSD adapter:** You' ll need to purchase a microSD to USB adapter so you can flash the memory card from your laptop.

If you don' t already have a Raspberry Pi 4, I highly recommend CanaKits ([which are available on Amazon](#)) and directly through [Canakit' s website](#).

Most of their kits come with a Raspberry Pi, power adapter, microSD, microSD adapter, heatsinks, and more!



**Figure 1:** Hardware for installing OpenCV 4 on your Raspberry Pi 4 running Raspbian Buster. Once you have the hardware ready, you' ll need to **flash a fresh copy of the Raspbian Buster operating system** to the microSD card.

1. Head on over to the [official BusterOS download page](#) (**Figure 2**), and start your download. I recommend the *"Raspbian Buster with Desktop and recommended software"* .
2. Download [Balena Etcher](#) — software for flashing memory cards. It works on every major OS.
3. Use Etcher to flash BusterOS to your memory card (**Figure 3**).




## Raspbian

**Raspbian** is the Foundation's official supported operating system. You can install it with [NOOBS](#) or download the image below and follow our [installation guide](#).

Raspbian comes pre-installed with plenty of software for education, programming and general use. It has Python, Scratch, Sonic Pi, Java and more.

The Raspbian with Desktop image contained in the ZIP archive is over 4GB in size, which means that these archives use features which are not supported by older unzip tools on some platforms. If you find that the download appears to be corrupt or the file is not unzipping correctly, please try using [7Zip](#) (Windows) or [The Unarchiver](#) (Macintosh). Both are free of charge and have been tested to unzip the image correctly.



### Raspbian Buster with desktop and recommended software


Image with desktop and recommended software based on Debian Buster

Version:	July 2019
Release date:	2019-07-10
Kernel version:	4.19
Size:	1945 MB

[Release notes](#)

[Download Torrent](#) [Download ZIP](#)

SHA-256: 2bd0613ec8739b6fa4274ea186ec859046f79e6aee4b8c6af0acb6d88f3f533a



### Raspbian Buster with desktop


Image with desktop based on Debian Buster

Version:	July 2019
Release date:	2019-07-10
Kernel version:	4.19
Size:	1149 MB

[Release notes](#)

[Download Torrent](#) [Download ZIP](#)

SHA-256: 6a1a5f20329e580d5161a0255b3d4163db6f56c3997e1c3b36bdd51140bd768e



### Raspbian Buster Lite

Minimal image based on Debian Buster

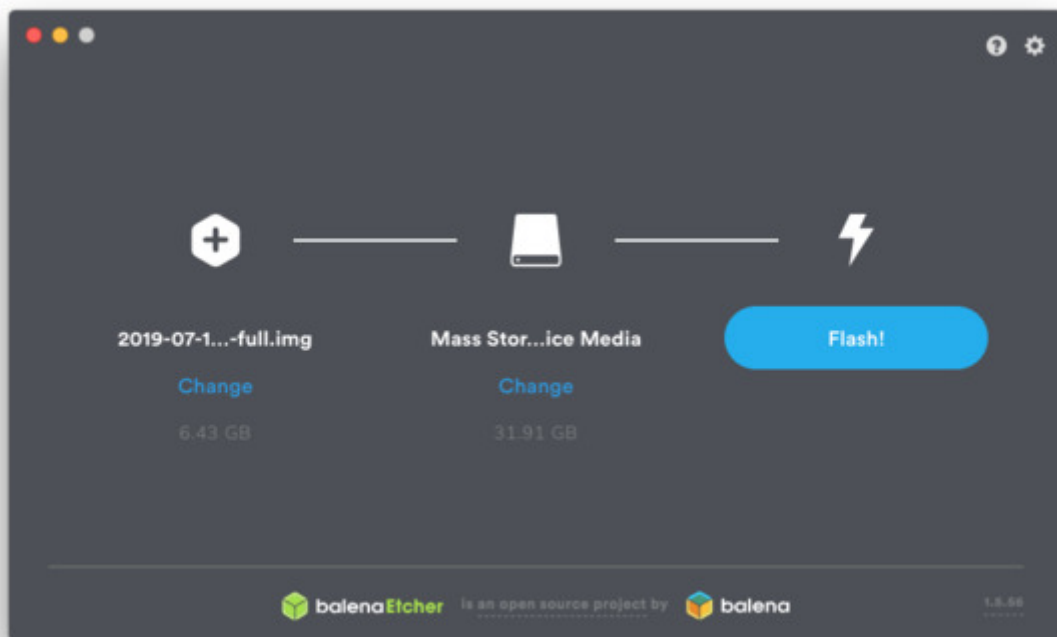
Version:	July 2019
Release date:	2019-07-10
Kernel version:	4.19
Size:	426 MB

[Release notes](#)

[Download Torrent](#) [Download ZIP](#)

SHA-256: 9e5cf24ce483bb96e7736ea75ca422e3560e7b455eee63dd28f66fa1825db70e

**Figure 2:** Download Raspbian Buster for your Raspberry Pi and OpenCV 4. After downloading the Raspbian Buster .img file you can flash it to your micro-SD card using Etcher:



**Figure 3:** Flash Raspbian Buster with Etcher. We will use BusterOS to install OpenCV 4 on our Raspberry Pi 4.

After a few minutes the flashing process should be complete — slot the micro-SD card into your Raspberry Pi 4 and then boot.

From there you can move on to the rest of the OpenCV install steps in this guide.

## Step #1: Expand filesystem and reclaim space

For the remainder of this tutorial I'll be making the following assumptions:

1. You are working with a brand new, fresh install of **Raspbian Buster** (see the previous section to learn how to flash Buster to your microSD).
2. You are comfortable with the command line and Unix environments.
3. You have an [SSH or VNC connection established with your Pi](#).

Alternatively, you could use a keyboard + mouse + screen.

Go ahead and insert your microSD into your Raspberry Pi and boot it up with a screen attached.

Once booted, configure your WiFi/ethernet settings to connect to the internet (you'll need an internet connection to download and install required packages for OpenCV).

From there you can [use SSH](#) as I have done, or go ahead and open a terminal.

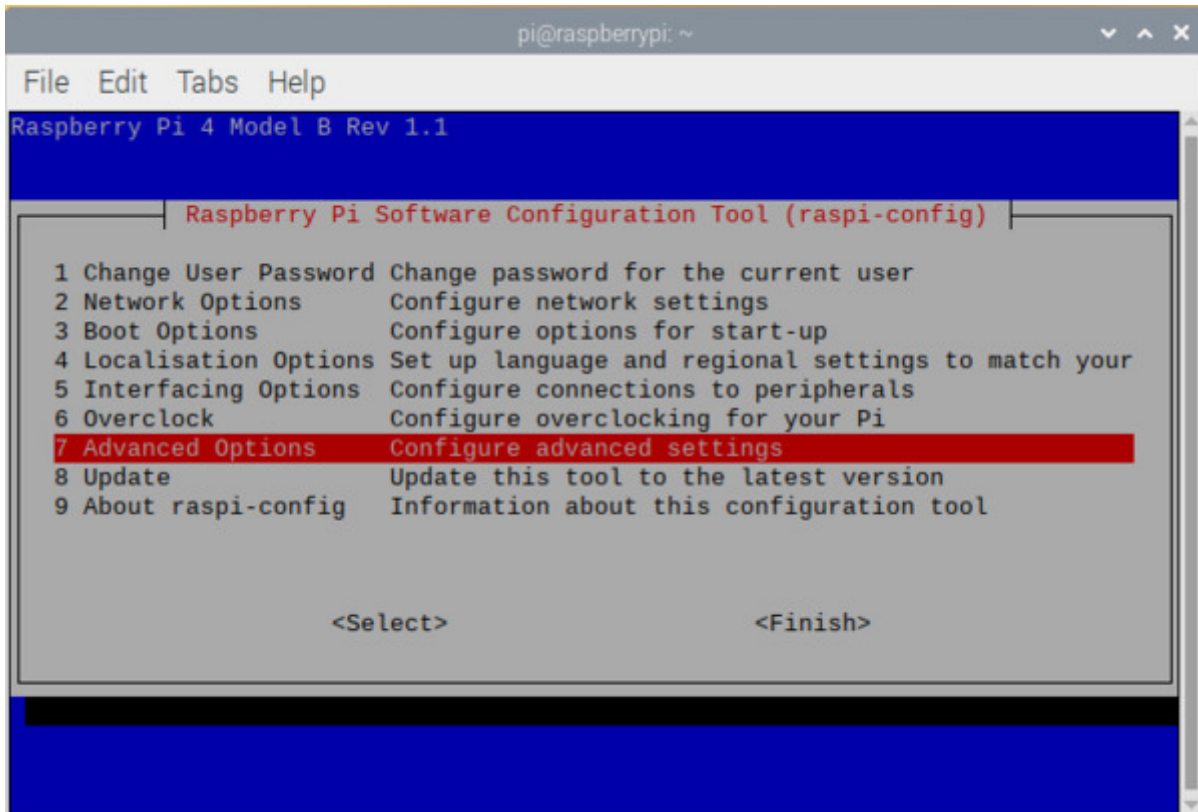
The first step is to run, `raspi-config` and expand your filesystem:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```
1 $ sudo raspi-config
```

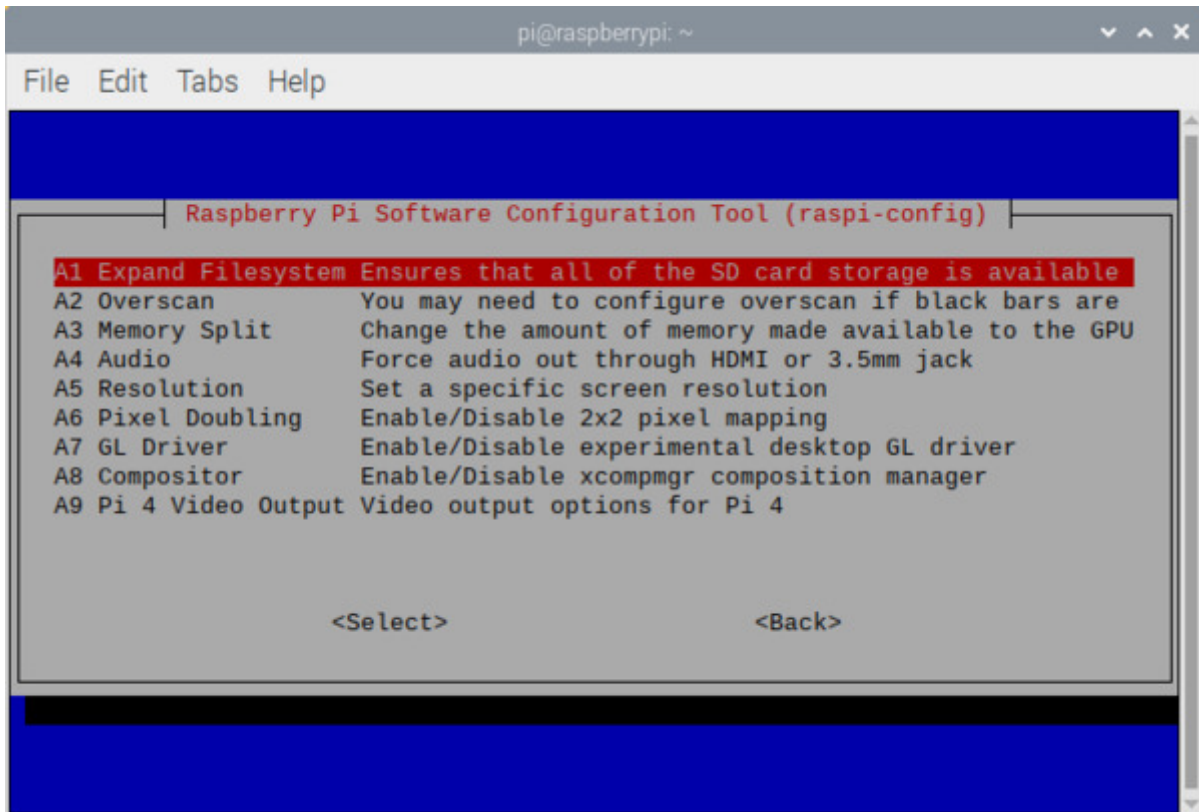
And then select the ***“7 Advanced Options”*** menu item:



**Figure 4:** The `raspi-config` configuration screen for Raspbian Buster. Select 7 Advanced Options so that we can expand our filesystem.

Followed by selecting ***“A1 Expand filesystem”*** :





**Figure 5:** The A1 Expand Filesystem menu item allows you to expand the filesystem on your microSD card containing the Raspberry Pi Buster operating system. Then we can proceed to install OpenCV 4.

Once prompted, you should select the first option, ***“A1 Expand File System”***, hit `enter` on your keyboard, arrow down to the ***“<Finish>”*** button, and then reboot your Pi — you may be prompted to reboot, but if you aren’ t you can execute:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```
1 $ sudo reboot
```

After rebooting, your file system should have been expanded to include all available space on your micro-SD card. You can verify that the disk has been expanded by executing `df -h` and examining the output:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```
1 $ df -h
2 Filesystem      Size  Used Avail Use% Mounted on
3 /dev/root        29G   5.3G   23G   20% /
4 devtmpfs        1.8G     0   1.8G    0% /dev
5 tmpfs           1.9G     0   1.9G    0% /dev/shm
6 tmpfs           1.9G   8.6M   1.9G    1% /run
7 tmpfs           5.0M   4.0K   5.0M    1% /run/lock
8 tmpfs           1.9G     0   1.9G    0% /sys/fs/cgroup
9 /dev/mmcblk0p1  253M   40M  213M   16% /boot
10 tmpfs           386M     0  386M    0% /run/user/1000
```

As you can see, my Raspbian filesystem has been expanded to include all 32GB of the micro-SD card.

However, even with my filesystem expanded, I have already used 15% of my 32GB card.

While it's not required, I would suggest deleting both Wolfram Engine and LibreOffice to reclaim ~1GB of space on your Raspberry Pi:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```
1 $ sudo apt-get purge wolfram-engine
2 $ sudo apt-get purge libreoffice*
3 $ sudo apt-get clean
4 $ sudo apt-get autoremove
```

## Step #2: Install dependencies

The following commands will update and upgrade any existing packages, followed by installing dependencies, I/O libraries, and optimization packages for OpenCV:

The first step is to update and upgrade any existing packages:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```
1 $ sudo apt-get update && sudo apt-get upgrade
```

We then need to install some developer tools, including [CMake](#), which helps us configure the OpenCV build process:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```
1 $ sudo apt-get install build-essential cmake pkg-config
```

Next, we need to install some image I/O packages that allow us to load various image file formats from disk. Examples of such file formats include JPEG, PNG, TIFF, etc.:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Python

```
1 $ sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng-dev
```

Just as we need image I/O packages, we also need video I/O packages. These libraries allow us to read various video file formats from disk as well as work



## directly with video streams:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```
1 $ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
2 $ sudo apt-get install libxvidcore-dev libx264-dev
```

The OpenCV library comes with a sub-module named `highgui` which is used to display images to our screen and build basic GUIs. In order to compile the `highgui` module, we need to install the GTK development library and prerequisites:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```
1 $ sudo apt-get install libfontconfig1-dev libcairo2-dev
2 $ sudo apt-get install libgdk-pixbuf2.0-dev libpango1.0-dev
3 $ sudo apt-get install libgtk2.0-dev libgtk-3-dev
```

Many operations inside of OpenCV (namely matrix operations) can be optimized further by installing a few extra dependencies:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```
1 $ sudo apt-get install libatlas-base-dev gfortran
```

These optimization libraries are *especially important* for resource-constrained devices such as the Raspberry Pi.

The following pre-requisites are for **Step #4a** and they certainly won't hurt for **Step #4b** either. They are for HDF5 datasets and Qt GUIs:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Python

```
1 $ sudo apt-get install libhdf5-dev libhdf5-serial-dev libhdf5-103
2 $ sudo apt-get install libqtgui4 libqtwebkit4 libqt4-test python3-pyqt5
```

Lastly, let's install Python 3 header files so we can compile OpenCV with Python bindings:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```
1 $ sudo apt-get install python3-dev
```

If you're working with a fresh install of the OS, it is possible that these versions of Python are already at the newest version (you'll see a terminal message stating this).

## Step #3: Create your Python virtual environment and install NumPy

We' ll be using Python **virtual environments**, a best practice when working with Python.

A Python virtual environment is an *isolated* development/testing/production environment on your system — it is fully sequestered from other environments. Best of all, you can manage the Python packages inside your your virtual environment inside with pip (Python' s package manager).

Of course, there are alternatives for managing virtual environments and packages (namely Anaconda/conda). I' ve used/tried them all, but have settled on pip, [virtualenv](#), and [virtualenvwrapper](#) as the preferred tools that I install on all of my systems. If you use the same tools as me, you' ll receive the best support from me.

You can install pip using the following commands:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```
1 $ wget https://bootstrap.pypa.io/get-pip.py
2 $ sudo python get-pip.py
3 $ sudo python3 get-pip.py
4 $ sudo rm -rf ~/.cache/pip
```

Let' s install `virtualenv` and `virtualenvwrapper` now:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```
1 $ sudo pip install virtualenv virtualenvwrapper
```

Once both `virtualenv` and `virtualenvwrapper` have been installed, open up your `~/.bashrc` file:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

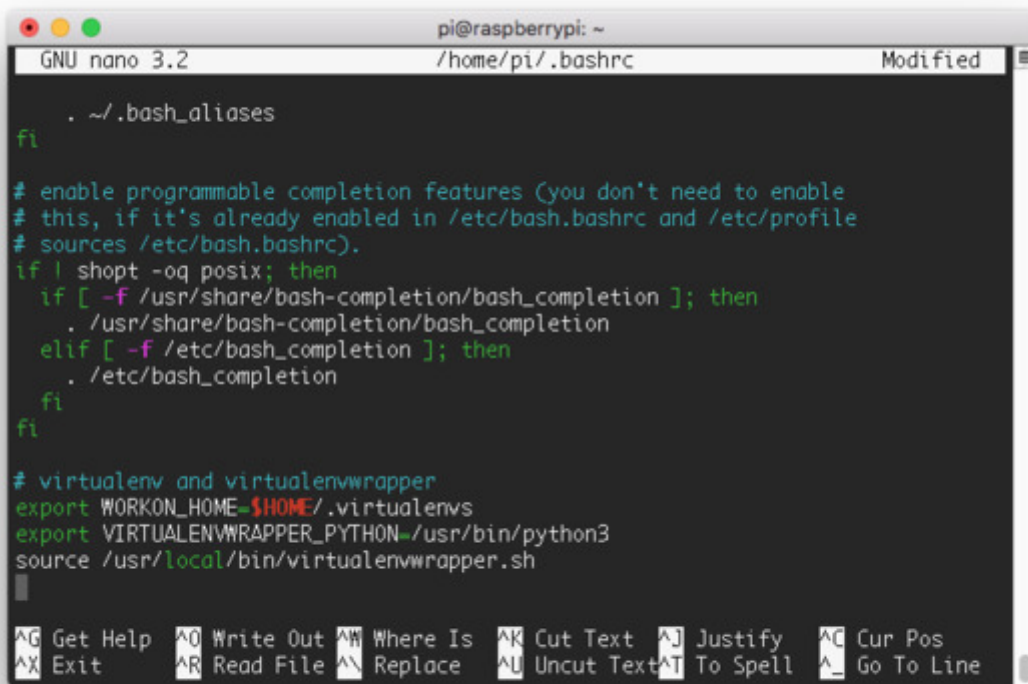
```
1 $ nano ~/.bashrc
```

...and append the following lines to the *bottom* of the file:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```
1 # virtualenv and virtualenvwrapper
2 export WORKON_HOME=$HOME/.virtualenvs
3 export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
4 source /usr/local/bin/virtualenvwrapper.sh
```



```
pi@raspberrypi: ~
GNU nano 3.2 /home/pi/.bashrc Modified

. ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

# virtualenv and virtualenvwrapper
export WORKON_HOME=$HOME/.virtualenvs
export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
source /usr/local/bin/virtualenvwrapper.sh
```

**Figure 6:** Using the nano editor to update ~/.bashrc with virtualenvwrapper settings.

Save and exit via `ctrl + x`, `y`, `enter`.

From there, reload your `~/.bashrc` file to apply the changes to your current bash session:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```
1 $ source ~/.bashrc
```

### Next, create your **Python 3 virtual environment**:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```
1 $ mkvirtualenv cv -p python3
```

Here we are creating a Python virtual environment named `cv` using Python 3. Going forward, I recommend Python 3 with OpenCV 4+.

**Note:** *Python 2.7 will reach end of its life on January 1st, 2020 so I **do not** recommend using Python 2.7.*

You can name the virtual environment whatever you want, but I use `cv` as the standard naming convention here on PyImageSearch.

If you have a Raspberry Pi Camera Module attached to your RPi, you should install the **PiCamera API** now as well:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```
1 $ pip install "picamera[array]"
```

## Step #4(a or b): Decide if you want the 1-minute quick install or the 2-hour complete install

From here you need to make a decision about the rest of your install. There are two options.

**1. Step #4a: pip install OpenCV 4:** If you decide to pip install OpenCV, you will be done **in a matter of seconds**. It is by far the *fastest, easiest* method to install OpenCV. It is the method I recommend for 90% of people — especially *beginners*. After this step, you will *skip to Step #5* to test your install.

**2. Step #4b: Compile OpenCV 4 from source:** This method gives you the **full install** of OpenCV 4. It will take 2-4 hours depending on the processor in your Raspberry Pi.

As stated, I highly encourage you to use the pip instructions. They are faster and will work for 90% of your projects. Additionally, the patented algorithms can only be used for educational purposes (there are plenty of great alternatives to the patented algorithms too).

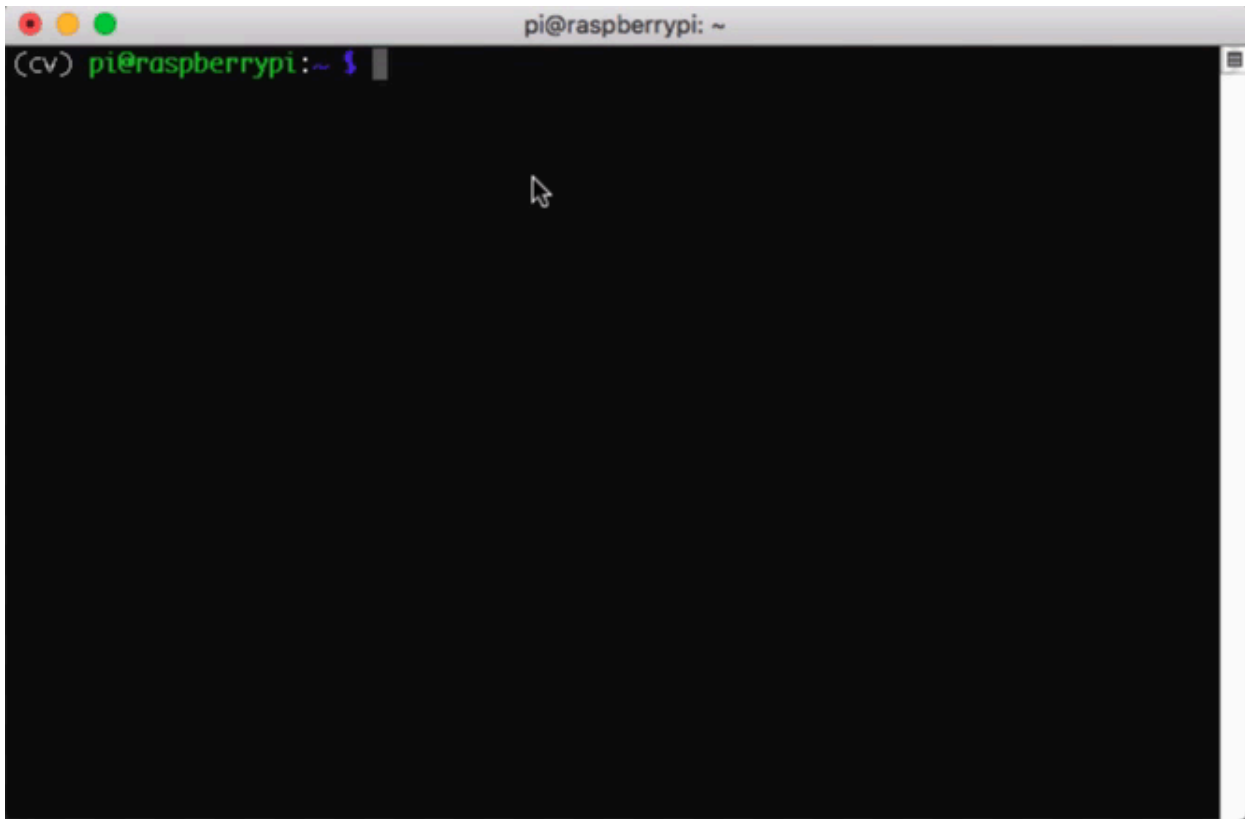
### Step #4a: pip install OpenCV 4

In a matter of seconds, you can **pip install OpenCV** into the `cv` virtual environment:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```
1 $ pip install opencv-contrib-python
```



**Figure 7:** To quickly install OpenCV 4 on your Raspberry Pi 4 running Raspbian Buster, I recommend using pip as shown.

If you watch the terminal output of the above screenshot carefully you'll see that OpenCV 3.4 rather than OpenCV 4 was installed?

What gives?

At the time of this writing, PiWheels has *not* been updated with pre-compiled OpenCV 4 binaries for Raspbian Buster. PiWheels normally lags slightly behind the latest version of OpenCV, likely to ensure compatibility across the major Raspbian releases. **Once OpenCV 4 has been released for PiWheels I will update this section.**

That's really all there is to it. You may skip to **Step #5** now to test your install.

## Step #4b: Compile OpenCV 4 from source

This option installs the full install of OpenCV including patented ( "Non-free" ) algorithms.

***Note: Do not follow Step #4b if you followed Step #4a.***

Let's go ahead and download the OpenCV source code for both the [opencv](#) and [opencv\\_contrib](#) repositories, followed by unarchiving them:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

---

```

1 $ cd ~
2 $ wget -O opencv.zip https://github.com/opencv/opencv/archive/4.1.1.zip
3 $ wget -O opencv_contrib.zip
4 https://github.com/opencv/opencv_contrib/archive/4.1.1.zip
5 $ unzip opencv.zip
6 $ unzip opencv_contrib.zip
7 $ mv opencv-4.1.1 opencv
$ mv opencv_contrib-4.1.1 opencv_contrib

```

For this blog post, we'll be using **OpenCV 4.1.1**; however, as newer versions of OpenCV are released you can update the corresponding version numbers.

## Increasing your SWAP space

Before you start the compile you must *increase your SWAP space*. Increasing the SWAP will enable you to compile OpenCV with *all four cores* of the Raspberry Pi (*and* without the compile hanging due to memory exhausting).

Go ahead and open up your `/etc/dphys-swapfile` file:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```

1 $ sudo nano /etc/dphys-swapfile

```

...and then edit the `CONF_SWAPSIZE` variable:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```

1 # set size to absolute value, leaving empty (default) then uses computed value
2 # you most likely don't want this, unless you have an special disk situation
3 # CONF_SWAPSIZE=100
4 CONF_SWAPSIZE=1024

```

Notice that I'm increasing the swap from 100MB to 1024MB. This is critical to compiling OpenCV with multiple cores on Raspbian Buster.

Save and exit via `ctrl + x`, `y`, `enter`.

***If you do not increase SWAP it's very likely that your Pi will hang during the compile.***

From there, restart the swap service:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```

1 $ sudo /etc/init.d/dphys-swapfile stop
2 $ sudo /etc/init.d/dphys-swapfile start

```

**Note:** Increasing swap size is a great way to burn out your Raspberry Pi microSD card. Flash-based storage has a limited number of writes you can perform until the card is essentially unable to hold the 1's and 0's



anymore. We' ll only be enabling large swap for a short period of time, so it' s not a big deal. Regardless, be sure to backup your `.img` file after installing OpenCV + Python just in case your card dies unexpectedly early. You can read more about large swap sizes corrupting memory cards on [this page](#).

## Compile and install OpenCV 4 on Raspbian Buster

We' re now ready to compile and install the full, optimized OpenCV library on the Raspberry Pi 4.

Ensure you are in the `cv` virtual environment using the `workon` command:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```
1 $ workon cv
```

Then, go ahead and install NumPy (an OpenCV dependency) into the Python virtual environment:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```
1 $ pip install numpy
```

And from there configure your build:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```
1 $ cd ~/opencv
2 $ mkdir build
3 $ cd build
4 $ cmake -D CMAKE_BUILD_TYPE=RELEASE \
5     -D CMAKE_INSTALL_PREFIX=/usr/local \
6     -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib/modules \
7     -D ENABLE_NEON=ON \
8     -D ENABLE_VFPV3=ON \
9     -D BUILD_TESTS=OFF \
10    -D INSTALL_PYTHON_EXAMPLES=OFF \
11    -D OPENCV_ENABLE_NONFREE=ON \
12    -D CMAKE_SHARED_LINKER_FLAGS=-latomic \
13    -D BUILD_EXAMPLES=OFF ..
```

There are four CMake flags I' d like to bring to your attention:

- **(1)** NEON and **(2)** VFPv3 optimization flags have been enabled. These lines ensure that you compile the fastest and most optimized OpenCV for the ARM processor on the Raspberry Pi (**Lines 7 and 8**).

- **Note:** The *Raspberry Pi Zero W* hardware is not compatible with NEON or VFPv3. Be sure to remove **Lines 7 and 8** if you are compiling for a *Raspberry Pi Zero W*.

- (3) Patented “NonFree” algorithms give you the full install of OpenCV (**Line 11**).
- And by drilling into OpenCV’s source, it was determined that we need the (4) `-latomic` shared linker flag (**Line 12**).

I’d like to take a second now to bring awareness to a **common pitfall** for beginners:

- In the terminal block above, you change directories into `~/opencv/`.
- You then create a `build/` directory therein and change directories into it.
- **If you try to execute CMake without being in the `~/opencv/build` directory, CMake will fail.** Try running `pwd` to see which working directory you are in *before* running `cmake`.

The `cmake` command will take about 3-5 minutes to run as it prepares and configures OpenCV for the compile.

When CMake finishes, be sure to inspect the output of CMake under the Python 3 section:

```
pi@raspberrypi: ~/opencv/build
-- Python 3:
--   Interpreter:      /home/pi/.virtualenvs/cv/bin/python3 (ver 3.
7.3)
--   Libraries:       /usr/lib/arm-linux-gnueabi/libpython3.7m.s
o (ver 3.7.3)
--   numpy:           /home/pi/.virtualenvs/cv/lib/python3.7/site-
packages/numpy/core/include (ver 1.17.2)
--   install path:    lib/python3.7/site-packages/cv2/python-3.7
--
-- Python (for build): /usr/bin/python2.7
--
-- Java:
--   ant:              NO
--   JNI:              NO
--   Java wrappers:    NO
--   Java tests:       NO
--
-- Install to:        /usr/local
-----
-- Configuring done
-- Generating done
-- Build files have been written to: /home/pi/opencv/build
(cv) pi@raspberrypi:~/opencv/build $
```

**Figure 8:** CMake configures your OpenCV 4 compilation from source on your Raspberry Pi 4 running Buster.

Notice how the `Interpreter` , `Libraries` , `numpy` , and `packages` path variables have been properly set. Each of these refers to our `cv` virtual environment. Now go ahead and **scroll up** to ensure that the **“Non-Free algorithms”** are set to be installed:

```
pi@raspberrypi: ~/opencv/build
-- OpenCV modules:
--   to be built:      aruco bgsegm bioinspired calib3d ccalib core
datasets dnn dnn_objdetect dpm face features2d flann freetype fuzzy gapi hdf hf
s highgui img_hash imgcodecs imgproc line_descriptor ml_objdetect optflow phase_
unwrapping photo plot python2 python3 quality reg rgbd saliency shape stereo sti
tching structured_light superres surface_matching text tracking ts video videoio
videostab xfeatures2d ximgproc xobjdetect xphoto
--   Disabled:        world
--   Disabled by dependency: -
--   Unavailable:     cnn_3dobj cudaarithm cudabgsegm cudacodec cu
dafeatures2d cudafilters cudaimgproc cudalegacy cudaobjdetect cudaoptflow cudast
ereo cudawarping cudev cvv java js matlab ovis sfm viz
--   Applications:    perf_tests apps
--   Documentation:    NO
-- Non-free algorithms: YES
--
GUI:
--   GTK+:             YES (ver 3.24.5)
--   GThread :         YES (ver 2.58.3)
--   GtkGLExt:         NO
--   VTK support:      NO
--
Media I/O:
--   ZLib:             /usr/lib/arm-linux-gnueabi/libz.so (ver 1.
```

**Figure 9:** Installing OpenCV 4 with “Non-free algorithms” on Raspbian Buster.

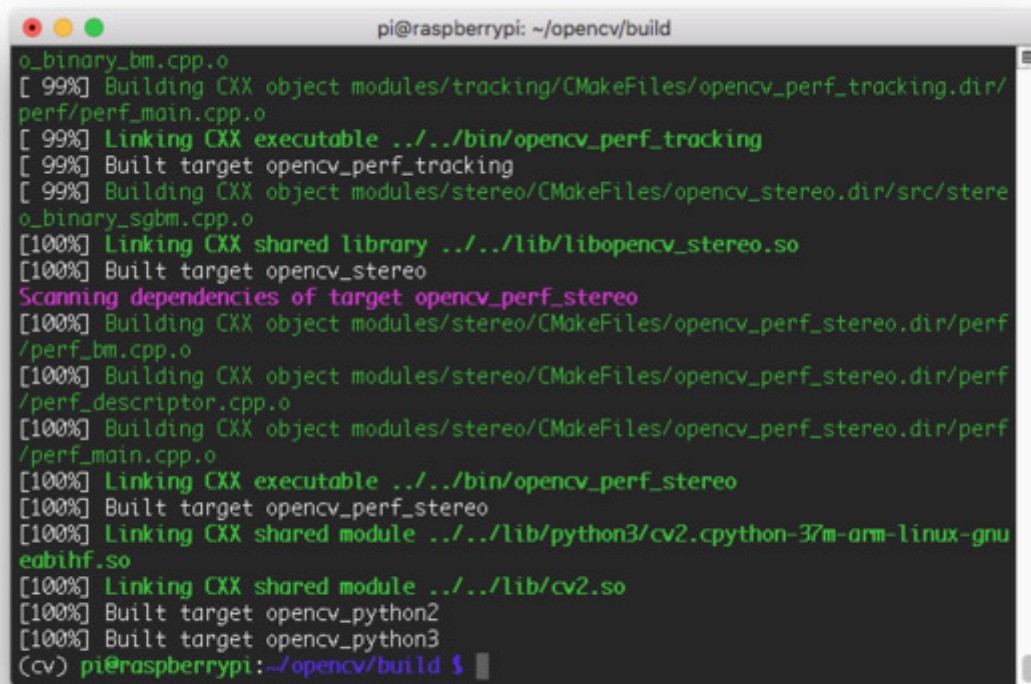
As you can see, “Non-free algorithms” for OpenCV 4 will be compiled + installed.

Now that we’ve prepared for our OpenCV 4 compilation, it is time to launch the compile process using all four cores:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```
1 $ make -j4
```



```
pi@raspberrypi: ~/opencv/build
o_binary_bm.cpp.o
[ 99%] Building CXX object modules/tracking/CMakeFiles/opencv_perf_tracking.dir/
perf/perf_main.cpp.o
[ 99%] Linking CXX executable ../../bin/opencv_perf_tracking
[ 99%] Built target opencv_perf_tracking
[ 99%] Building CXX object modules/stereo/CMakeFiles/opencv_stereo.dir/src/stere
o_binary_sgbm.cpp.o
[100%] Linking CXX shared library ../../lib/libopencv_stereo.so
[100%] Built target opencv_stereo
Scanning dependencies of target opencv_perf_stereo
[100%] Building CXX object modules/stereo/CMakeFiles/opencv_perf_stereo.dir/perf
/perf_bm.cpp.o
[100%] Building CXX object modules/stereo/CMakeFiles/opencv_perf_stereo.dir/perf
/perf_descriptor.cpp.o
[100%] Building CXX object modules/stereo/CMakeFiles/opencv_perf_stereo.dir/perf
/perf_main.cpp.o
[100%] Linking CXX executable ../../bin/opencv_perf_stereo
[100%] Built target opencv_perf_stereo
[100%] Linking CXX shared module ../../lib/python3/cv2.cpython-37m-arm-linux-gnu
eabihf.so
[100%] Linking CXX shared module ../../lib/cv2.so
[100%] Built target opencv_python2
[100%] Built target opencv_python3
(cv) pi@raspberrypi:~/opencv/build $
```

**Figure 10:** We used Make to compile OpenCV 4 on a Raspberry Pi 4 running Raspbian Buster. Running `make` could take anywhere from 1-4 hours depending on your Raspberry Pi hardware (this tutorial is compatible with the Raspberry Pi 3B, 3B+, and 4). The Raspberry Pi 4 is the fastest at the time of this writing. Assuming OpenCV compiled without error (as in my screenshot above), you can install your optimized version of OpenCV on your Raspberry Pi:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```
1 $ sudo make install
2 $ sudo ldconfig
```

## Reset your SWAP

***Don't forget to go back*** to your `/etc/dphys-swapfile` file and:

1. Reset `CONF_SWAPSIZE` to 100MB.
2. Restart the swap service.

## Sym-link your OpenCV 4 on the Raspberry Pi

Symbolic links are a way of pointing from one directory to a file or folder elsewhere on your system. For this sub-step, we will sym-link the `cv2.so` bindings into your `cv` virtual environment.

Let's proceed to create our sym-link. Be sure to [use "tab-completion"](#) for all paths below (rather than copying these commands blindly):

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

```
1 $ cd /usr/local/lib/python3.7/site-packages/cv2/python-3.7
2 $ sudo mv cv2.cpython-37m-arm-linux-gnueabi.hf.so cv2.so
3 $ cd ~/.virtualenvs/cv/lib/python3.7/site-packages/
4 $ ln -s /usr/local/lib/python3.7/site-packages/cv2/python-3.7/cv2.so cv2.so
```

Keep in mind that the exact paths may change and you should [use "tab-completion"](#).

## Step 5: Testing your OpenCV 4 Raspberry Pi BusterOS install

As a quick sanity check, access the `cv` virtual environment, fire up a Python shell, and try to import the OpenCV library:

Install OpenCV4 on Raspberry Pi 4 and Raspbian Buster

Shell

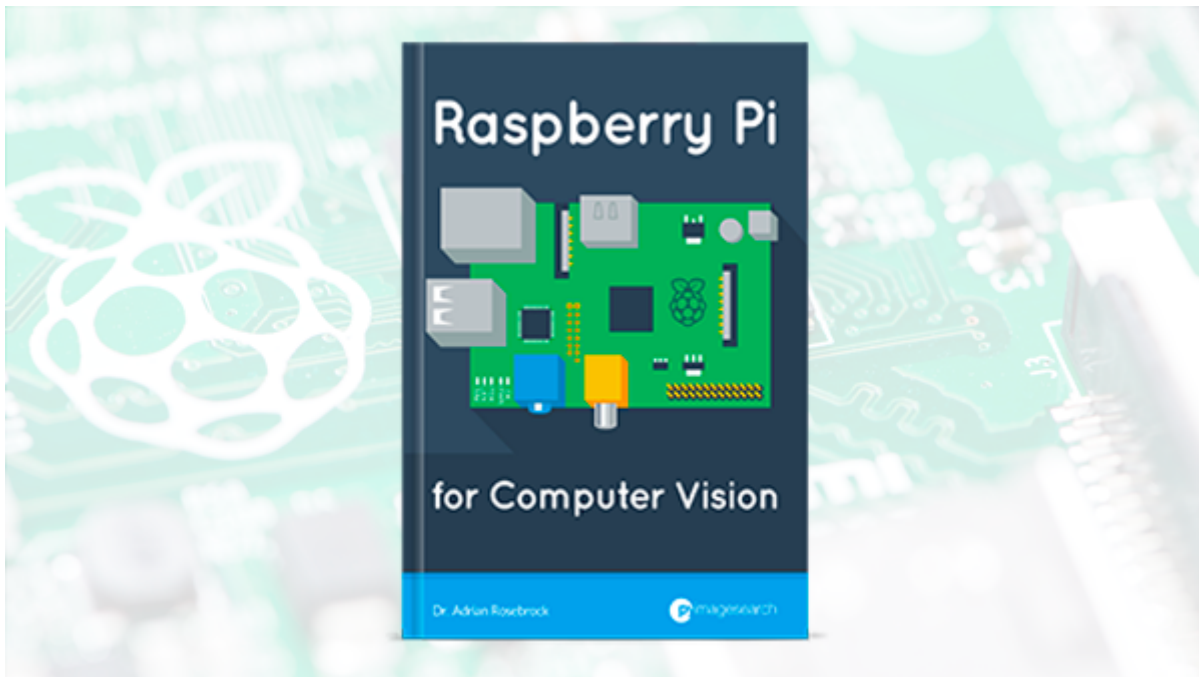
```
1 $ cd ~
2 $ workon cv
3 $ python
4 >>> import cv2
5 >>> cv2.__version__
6 '4.1.1'
7 >>>
```

**Congratulations!** You've just installed an OpenCV 4 on your Raspberry Pi.

If you are looking for some fun projects to work on with OpenCV 4, be sure to checkout my [Raspberry Pi archives](#).

## What's next?





Ready to put your Raspberry Pi and OpenCV install to work?

My brand new book, [\*Raspberry Pi for Computer Vision\*](#), has **over 40 Computer Vision and Deep Learning projects** for embedded computer vision and Internet of Things (IoT) applications. You can build upon the projects in the book to solve problems around your home, business, and even for your clients. **Each of these projects place an emphasis on:**

- Learning by doing
- Rolling up your sleeves
- Getting your hands dirty in code and implementation
- Building *actual, real-world projects* using the Raspberry Pi

**A handful of the highlighted projects include:**

- Daytime and nighttime wildlife monitoring
- Traffic counting and vehicle speed detection
- Deep Learning classification, object detection, and instance segmentation on resource constrained devices
- Hand gesture recognition
- Basic robot navigation
- Security applications
- Classroom attendance
- *...and many more!*

The book also covers deep learning using the **Google Coral** and **Intel Movidius NCS** coprocessors along with **NVIDIA Jetson Nano** board. If you' re interested in studying Computer Vision and Deep Learning on embedded devices, [\*you won' t find a better book than this one!\*](#)

**Pick up your copy of** \_\_\_\_\_ **today!**

## Summary

In today' s tutorial, you learned how to install OpenCV 4 on your Raspberry Pi 4 running the Raspbian Buster operating system via two methods:

- A simple **pip install** (fast and easy)
- **Compiling from source** (takes longer, but gives you the full OpenCV install/optimizations)

The pip method to install OpenCV 4 is *by far the easiest way* to install OpenCV (and the method I *recommend* for 90% of projects). It is especially great for beginners too.

If you need the full install of OpenCV, you must compile from source.

Compiling from source ensures that you have the full install including the "contrib" module with patented ( "NonFree" ) algorithms.

**While compiling from source is both (1) more complicated, and (2) more time-consuming, it is currently the only way to access all features of OpenCV.**

I hope you enjoyed today' s tutorial!

And if you' re ready to put your RPi and OpenCV install to work, be sure to check out my book, [\*Raspberry Pi for Computer Vision\*](#) — inside the book you' ll learn how to build practical, real-world Computer Vision and Deep Learning applications on the Raspberry Pi, Google Coral, Movidius NCS, and NVIDIA Jetson Nano.