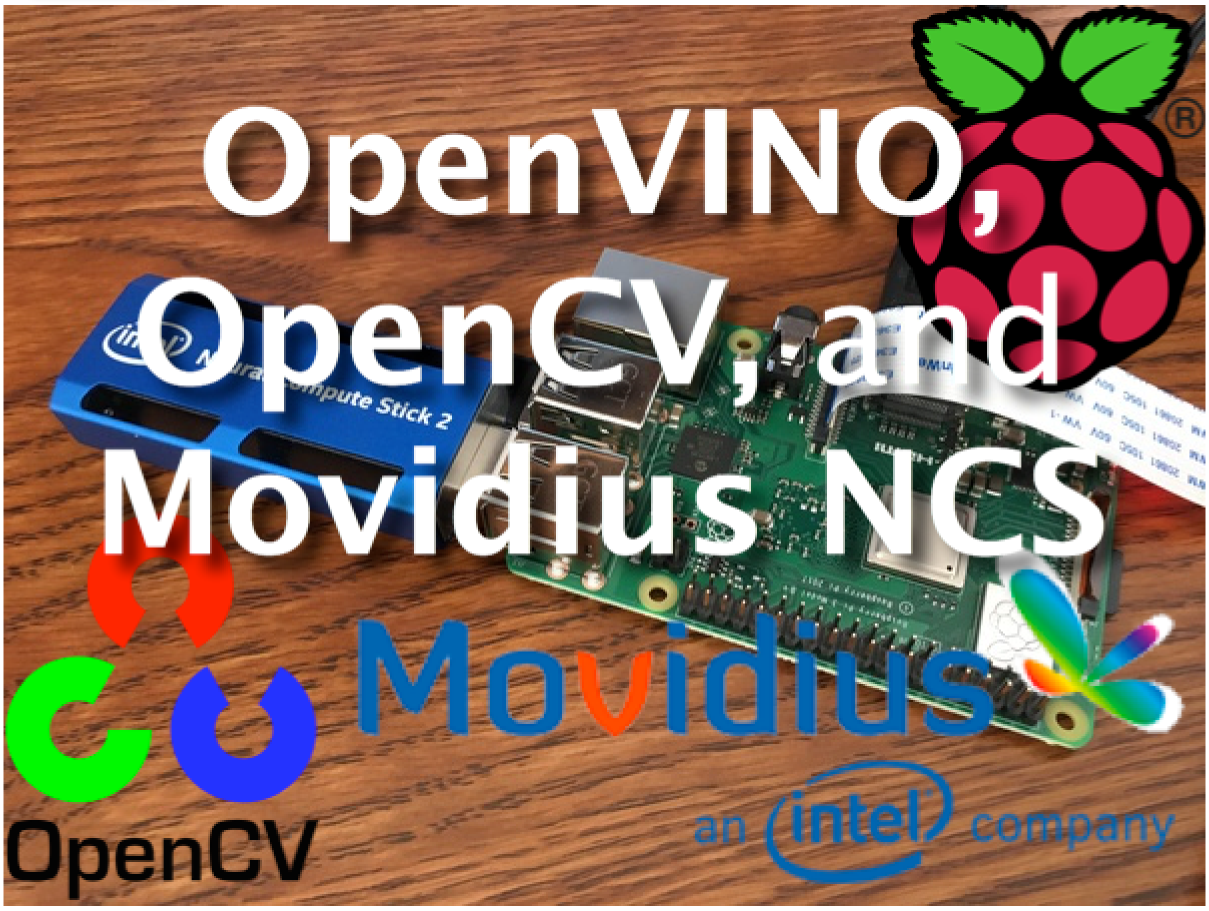
OpenVINO, OpenCV, and Movidius NCS on the Raspberry Pi

by Adrian Rosebrock on April 8, 2019 in Deep Learning, Embedded, IoT, Movidius, Raspberry Pi, Tutorials

Click here to download the source code to this post



Inside this tutorial, you will learn how to utilize the OpenVINO toolkit with OpenCV for faster deep learning inference on the Raspberry Pi.

Raspberry Pis are great — I love the quality hardware and the supportive community built around the device.

That said, for deep learning, the current Raspberry Pi hardware is inherently resource-constrained and you' ll be lucky to get more than a few FPS (using the RPi CPU alone) out of most state-of-the-art models (especially object detection and instance/semantic segmentation).

We know from my previous posts that Intel' s Movidius Neural Compute Stick allows for faster inference with the deep learning coprocessor that you plug into the USB socket:

- *Getting started with the Intel Movidius Neural Compute Stick*
- *Real-time object detection on the Raspberry Pi with the Movidius NCS*

Since 2017, the Movidius team has been hard at work on their Myriad processors and their consumer grade USB deep learning sticks.

The first version of the API that came with the sticks worked well and demonstrated the power of the Myriad, but left a lot to be desired.

Then, the Movidius APIv2 was released and welcomed by the Movidius + Raspberry Pi community. It was easier/more reliable than the APIv1 but had its fair share of issues as well.

But now, it' s become *easier than ever* to work with the Movidius NCS, *especially* with OpenCV.

Meet OpenVINO, an Intel library for hardware optimized computer vision designed to replace the V1 and V2 APIs.

Intel' s shift to support the Movidius hardware with OpenVINO software makes the Movidius shine in all of its metallic blue glory.

OpenVINO is extremely simple to use — just set the target processor (a single function call) and let OpenVINO-optimized OpenCV handle the rest.

But the question remains:

How can I install OpenVINO on the Raspberry Pi?

Today we' ll learn just that, along with a practical object detection demo (spoiler alert: it is dead simple to use the Movidius coprocessor now).

To learn how to install OpenVINO on the Raspberry Pi (and perform object detection with the Movidius Neural Compute Stick), *just follow this tutorial!*

Looking for the source code to this post?

Jump right to the downloads section.

OpenVINO, OpenCV, and Movidius NCS on the Raspberry Pi

In this blog post we’ re going to cover three main topics.

1. First, we’ ll learn what OpenVINO is and how it is a very welcome paradigm shift for the Raspberry Pi.
2. We’ ll then cover how to install OpenCV and OpenVINO on your Raspberry Pi.
3. Finally, we’ ll develop a real-time object detection script using OpenVINO, OpenCV, and the Movidius NCS.

***Note:** There are many Raspberry Pi install guides on my blog, most unrelated to Movidius. Before you begin, be sure to check out the available install tutorials on my **OpenCV installation guides** page and choose the one that best fits your needs.*

Let’ s get started.

What is OpenVINO?

INTEL® DISTRIBUTION OF OPENVINO™ TOOLKIT

DEVELOP MULTIPLATFORM COMPUTER VISION SOLUTIONS

Explore the Intel® Distribution of OpenVINO™ toolkit

Make your vision a reality on Intel® platforms—from smart cameras and video surveillance to robotics, transportation, and more.

Your Computer Vision Apps...Now Faster

Develop applications and solutions that emulate human vision with the Intel® Distribution of OpenVINO™ toolkit. Based on convolutional neural networks (CNN), the toolkit extends workloads across Intel® hardware (including accelerators) and maximizes performance.

- Enables CNN-based deep learning inference at the edge
- Supports heterogeneous execution across computer vision accelerators—CPU, GPU, Intel® Movidius™ Neural Compute Stick, and FPGA—using a common API
- Speeds up time to market via a library of functions and preoptimized kernels
- Includes optimized calls for OpenCV and OpenVX*

Get Started
Innovate Vision Solutions with OpenVINO...
Intel® Distribution of OpenVINO™ toolkit overview...
Share

DISCOVER THE CAPABILITIES

Cat ✕
Cat ✓
GPU
CPU
Less System Workload
Deep Learning for Computer Vision
Traditional Computer Vision
Hardware Acceleration

Figure 1: The Intel OpenVINO toolkit optimizes your computer vision apps for Intel hardware such as the Movidius Neural Compute Stick. Real-time object detection with OpenVINO and OpenCV using Raspberry Pi and Movidius NCS sees a significant speedup. (source)

Intel’ s OpenVINO is an acceleration library for optimized computing with Intel’ s hardware portfolio.

OpenVINO supports Intel CPUs, GPUs, FPGAs, and VPUs.

Deep learning libraries you’ ve come to rely upon such as TensorFlow, Caffe, and mxnet are supported by OpenVINO.

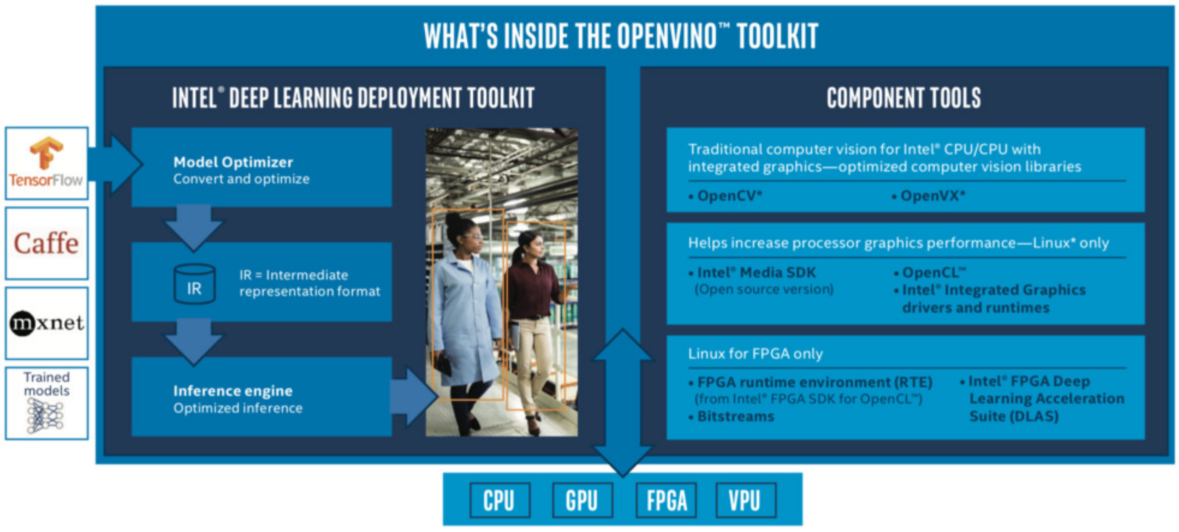

WHAT'S INSIDE THE OPENVINO™ TOOLKIT
INTEL® DEEP LEARNING DEPLOYMENT TOOLKIT
COMPONENT TOOLS
TensorFlow
Caffe
mxnet
Trained models
Model Optimizer
Convert and optimize
IR
IR = Intermediate representation format
Inference engine
Optimized inference
Traditional computer vision for Intel® CPU/CPU with integrated graphics—optimized computer vision libraries
• OpenCV*
• OpenVX*
Helps increase processor graphics performance—Linux* only
• Intel® Media SDK (Open source version)
• OpenCL™
• Intel® Integrated Graphics drivers and runtimes
Linux for FPGA only
• FPGA runtime environment (RTE) (from Intel® FPGA SDK for OpenCL™)
• Bitstreams
• Intel® FPGA Deep Learning Acceleration Suite (DLAS)
CPU
GPU
FPGA
VPU

Figure 2: The Intel OpenVINO Toolkit supports intel CPUs, GPUs, FPGAs, and VPUs. TensorFlow, Caffe, mxnet, and OpenCV' s DNN module all are optimized and accelerated for Intel hardware. The Movidius line of vision processing units (VPUs) are supported by OpenVINO and pair well with the Raspberry Pi. (source: OpenVINO Product Brief)

Intel has even optimized OpenCV' s DNN module to support its hardware for deep learning.

In fact, many newer smart cameras use Intel' s hardware along with the OpenVINO toolkit. OpenVINO is **edge computing** and **IoT** at its finest — it enables resource-constrained devices like the Raspberry Pi to work with the Movidius coprocessor to perform deep learning at speeds that are useful for real-world applications.

We' ll be installing OpenVINO on the Raspberry Pi so it can be used with the Movidius VPU (Vision Processing Unit) in the next section.

Be sure to read the OpenVINO product brief PDF for more information.

Installing OpenVINO' s optimized OpenCV on the Raspberry Pi

In this section, we' ll cover prerequisites and all steps required to install OpenCV and OpenVINO on your Raspberry Pi.

Be sure to read this entire section before you begin so that you are familiar with the steps required.

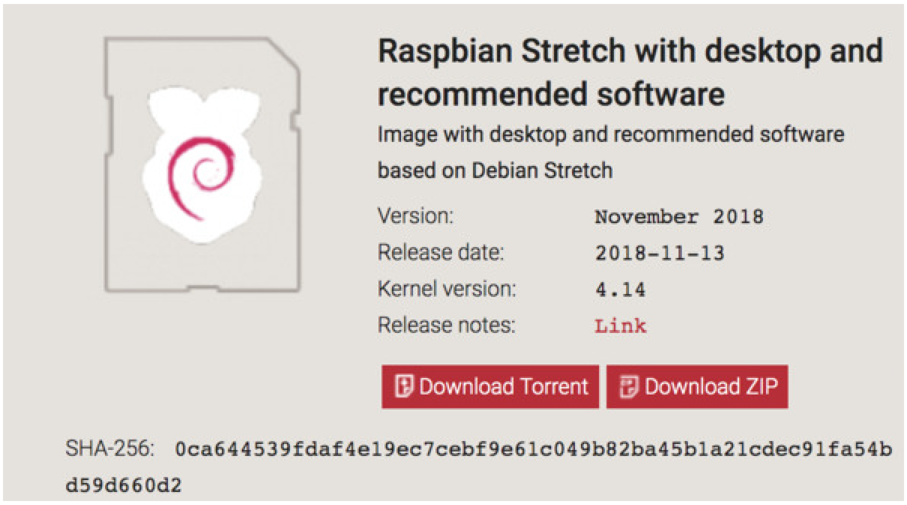
Let' s begin.

Hardware, assumptions, and prerequisites

In this tutorial, I am going to assume that you have the following hardware:

- **Raspberry Pi 3B+** (or Raspberry Pi 3B)
- **Movidius NCS 2** (or Movidius NCS 1)
- **PiCamera V2** (or USB webcam)
- **32GB microSD card *with* Raspbian Stretch freshly flashed** (16GB would likely work as well)
- **HDMI screen + keyboard/mouse** (at least for the initial WiFi configuration)
- **5V power supply** (I recommend a 2.5A supply because the Movidius NCS is a power hog)

If you don' t have a microSD with a fresh burn of Raspbian Stretch, you may download it here. I recommend the full install:



Raspbian Stretch with desktop and recommended software

Image with desktop and recommended software based on Debian Stretch

Version: November 2018
Release date: 2018-11-13
Kernel version: 4.14
Release notes: Link

SHA-256: 0ca644539fdaf4e19ec7cebf9e61c049b82ba45b1a21cdec91fa54bd59d660d2

Figure 3: The Raspbian Stretch operating system is required for OpenVINO and the Movidius on the Raspberry Pi.

From there, use Etcher (or a suitable alternative) to flash the card.

Once you' re ready, insert the microSD card into your Raspberry Pi and boot it up.

Enter your WiFi credentials and enable SSH, VNC, and the camera interface.

From here you will need one of the following:

- ***Physical access*** to your Raspberry Pi so that you can open up a terminal and execute commands
- ***Remote access*** via SSH or VNC

I' ll be doing the majority of this tutorial via SSH, but as long as you have access to a terminal, you can easily follow along.

Can' t SSH? If you see your Pi on your network, but can' t ssh to it, you may need to enable SSH. This can easily be done via the Raspberry Pi desktop preferences menu or using the `raspi-config` command.

After you' ve changed the setting and rebooted, you can test SSH directly on the Pi with the localhost address.

Open a terminal and type `ssh pi@127.0.0.1` to see if it is working. To SSH from another computer you' ll need the Pi' s IP address — you can determine the IP address by looking at your router' s clients page or by running `ifconfig` to determine the IP on/of the Pi itself.

Is your Raspberry Pi keyboard layout giving you problems? Change your keyboard layout by going to the Raspberry Pi desktop preferences menu. I use the standard US Keyboard layout, but you' ll want to select the one appropriate for you.

Step #0: Expand filesystem on your Raspberry Pi

To get the OpenVINO party started, fire up your Raspberry Pi and open an SSH connection (alternatively use the Raspbian desktop with a keyboard + mouse and launch a terminal).

If you' ve just flashed Raspbian Stretch, I always recommend that you first check to ensure your filesystem is using all available space on the microSD card.

To check your disk space usage execute the `df -h` command in your terminal and examine the output:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

```
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        30G  4.2G   24G  15% /
devtmpfs        434M     0  434M   0% /dev
tmpfs           438M     0  438M   0% /dev/shm
tmpfs           438M   12M  427M   3% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           438M     0  438M   0% /sys/fs/cgroup
/dev/mmcblk0p1   42M   21M   21M  51% /boot
tmpfs            88M     0   88M   0% /run/user/1000
```

As you can see, my Raspbian filesystem has been automatically expanded to include all 32GB of the micro-SD card. This is denoted by the fact that the size is 30GB (nearly 32GB) and I have 24GB available (15% usage).

If you' re seeing that you aren' t using your entire memory card capacity, below you can find instructions on how to expand the

filesystem.

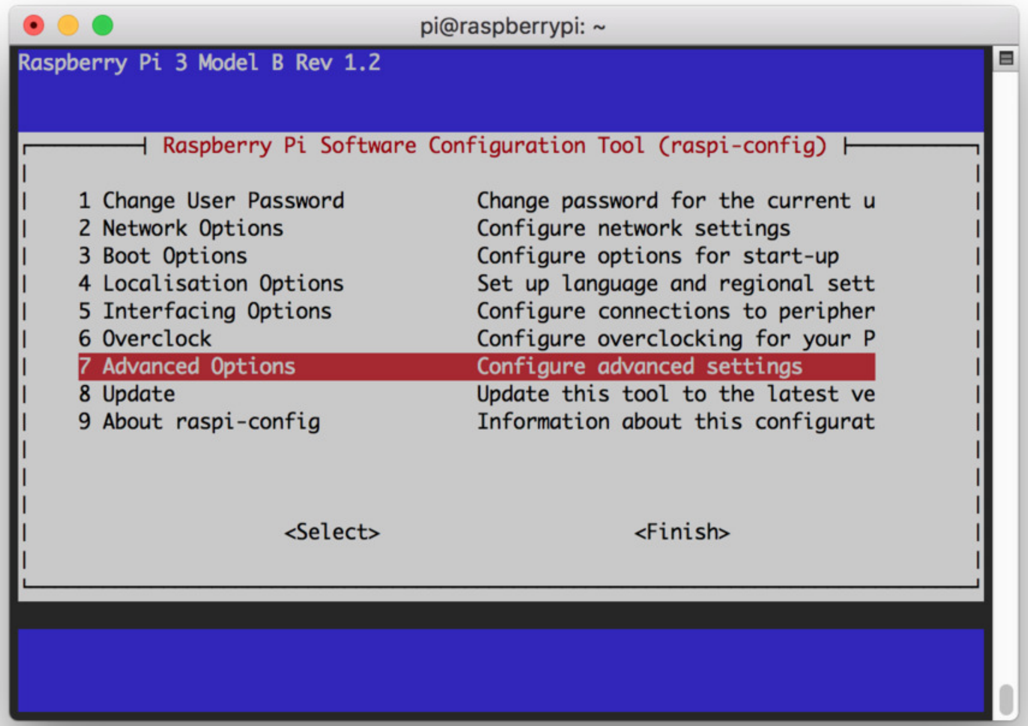
Open up the Raspberry Pi configuration in your terminal:

```
Installing OpenVINO on the Raspberry Pi and performing object detection with the
Movidius Neural Compute Stick
```

Shell

```
$ sudo raspi-config
```

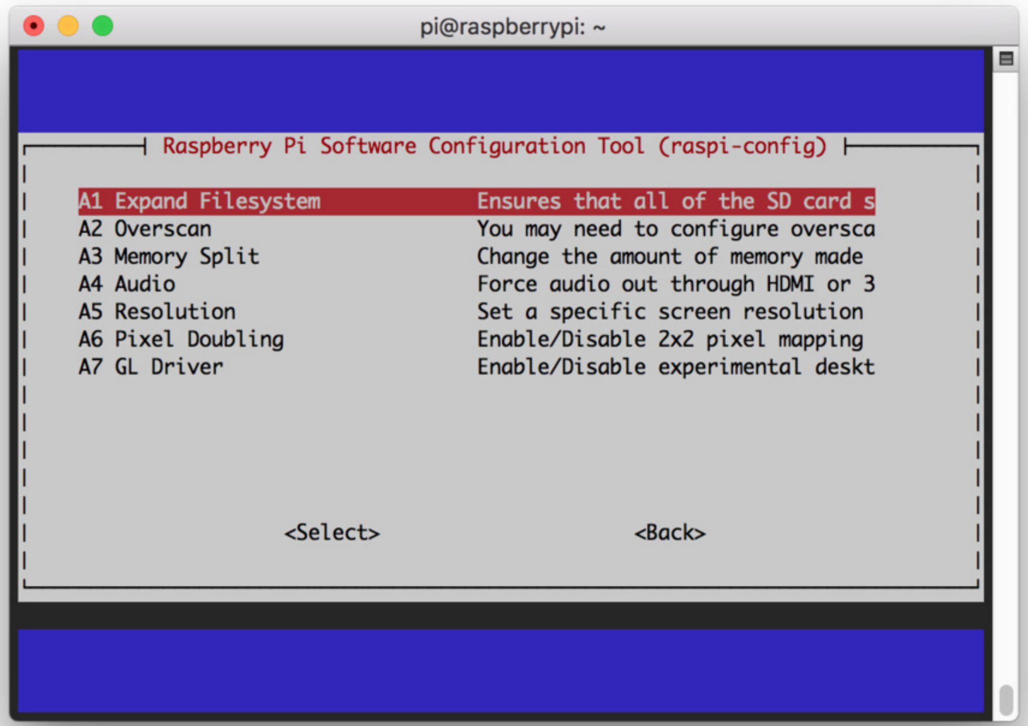
And then select the *"Advanced Options"* menu item:



```
pi@raspberrypi: ~
Raspberry Pi 3 Model B Rev 1.2

┤ Raspberry Pi Software Configuration Tool (raspi-config) ├

   1 Change User Password          Change password for the current u
   2 Network Options               Configure network settings
   3 Boot Options                  Configure options for start-up
   4 Localisation Options          Set up language and regional sett
   5 Interfacing Options           Configure connections to peripher
   6 Overclock                     Configure overclocking for your P
   7 Advanced Options              Configure advanced settings
   8 Update                        Update this tool to the latest ve
   9 About raspi-config            Information about this configurat

              <Select>                       <Finish>
```

Figure 4: Selecting the "Advanced Options" from the `raspi-config` menu to expand the Raspbian file system on your Raspberry Pi is important before installing OpenVINO and OpenCV. Next, we' ll actually expand the filesystem.

Followed by selecting *"Expand filesystem"* :



```
pi@raspberrypi: ~

Raspberry Pi Software Configuration Tool (raspi-config)

A1 Expand Filesystem          Ensures that all of the SD card s
A2 Overscan                   You may need to configure oversca
A3 Memory Split               Change the amount of memory made
A4 Audio                      Force audio out through HDMI or 3
A5 Resolution                 Set a specific screen resolution
A6 Pixel Doubling             Enable/Disable 2x2 pixel mapping
A7 GL Driver                  Enable/Disable experimental deskt

              <Select>                      <Back>
```

Figure 5: The Raspberry Pi "Expand Filesystem" menu allows us to take advantage of our entire flash memory card. This will give us the space necessary to install OpenVINO, OpenCV, and other packages.

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 will be prompted to reboot. Alternatively, you can reboot from the terminal:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

```
$ sudo reboot
```

Be sure to run the `df -h` command again to check that your file system is expanded.

Step #1: Reclaim space on your Raspberry Pi

One simple way to gain more space on your Raspberry Pi is to delete both LibreOffice and Wolfram engine to free up some space on your Pi:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

```
$ sudo apt-get purge wolfram-engine
$ sudo apt-get purge libreoffice*
$ sudo apt-get clean
$ sudo apt-get autoremove
```

After removing the Wolfram Engine and LibreOffice, **you can reclaim almost 1GB!**

Step #3: Install OpenVINO + OpenCV dependencies on your Raspberry Pi

This step shows some dependencies which I install on every OpenCV system. While you'll soon see that OpenVINO is already compiled, I recommend that you go ahead and install these packages anyway in case you end up compiling OpenCV from scratch at any time going forward.

Let's update our system:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

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

And then install developer tools including CMake:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

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

Next, it is time to install a selection of image and video libraries — these are *key* to being able to work with image and video files:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

```
$ sudo apt-get install libjpeg-dev libpng-dev libtiff-dev
$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
$ sudo apt-get install libxvidcore-dev libx264-dev
```

From there, let's install GTK, our GUI backend:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

```
$ sudo apt-get install libgtk-3-dev
```

And now let's install a package which may help to reduce GTK warnings:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

```
$ sudo apt-get install libcanberra-gtk*
```

The asterisk ensures we will grab the ARM-specific GTK. It is required.

Now we need two packages which contain numerical optimizations for OpenCV:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

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

And finally, let’ s install the Python 3 development headers:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

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

Once you have all of these prerequisites installed you can move on to the next step.

Step #4: Download and unpack OpenVINO for your Raspberry Pi

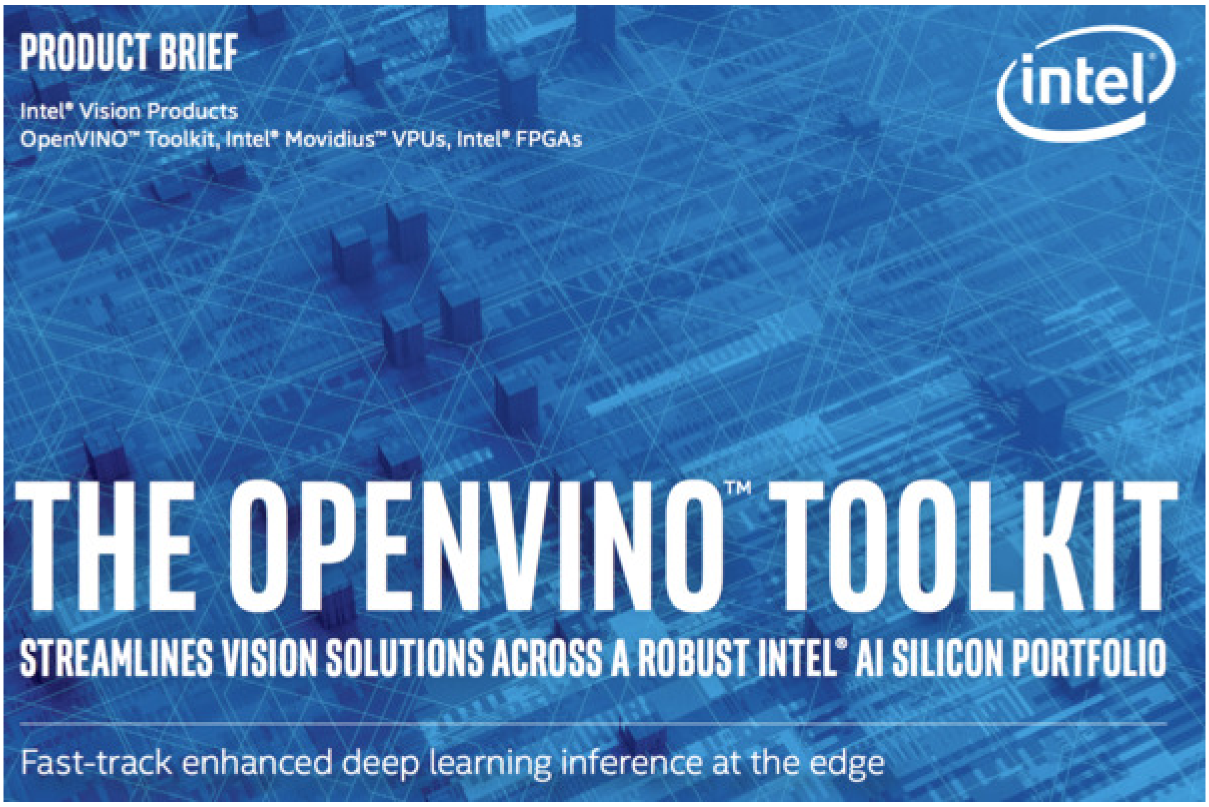

PRODUCT BRIEF
Intel® Vision Products
OpenVINO™ Toolkit, Intel® Movidius™ VPUs, Intel® FPGAs
intel
THE OPENVINO™ TOOLKIT
STREAMLINES VISION SOLUTIONS ACROSS A ROBUST INTEL® AI SILICON PORTFOLIO
Fast-track enhanced deep learning inference at the edge

Figure 6: Download and install the OpenVINO toolkit for Raspberry Pi and Movidius computer vision apps (source: Intel' s OpenVINO Product Brief).

From here forward, our install instructions are largely based upon Intel' s Raspberry Pi OpenVINO guide. There are a few "gotchas" which is why I decided to write a guide. We' ll also use virtual environments as PyImageSearch readers have come to expect.

Our next step is to download OpenVINO.

Let' s navigate to our home folder and create a new directory

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

```
$ cd ~
$ mkdir openvino
$ cd openvino
```

From there, go ahead and grab the OpenVINO toolkit for the Raspberry Pi download. You may try wget as I have, just beware of the problem noted in the subsequent codeblock:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

```
$ wget
http://download.01.org/openvinotoolkit/2018_R5/packages/l_openvino_toolkit_ie_p_20
5.445.tgz
```

At this point, through trial and error, I found that `wget` actually only grabbed an HTML file which seems to be a really strange server error at Intel' s download site.

Ensure that you actually have a tar file using this command:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Python

```
$ file l_openvino_toolkit_ie_p_2018.5.445.tgz

# bad output
l_openvino_toolkit_ie_p_2018.5.445.tgz: HTML document text, UTF-8 Unicode text, wi
very long lines

# good output
l_openvino_toolkit_ie_p_2018.5.445.tgz: gzip compressed data, was
"l_openvino_toolkit_ie_p_2018.5.445.tar", last modified: Wed Dec 19 12:49:53 2018,
max compression, from FAT filesystem (MS-DOS, OS/2, NT)
```

If the output matches the highlighted "good output" , then you can safely proceed to extract the archive. Otherwise, remove the file and try again.

Once you have successfully downloaded the OpenVINO toolkit, you can unarchive it using the following command:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

```
$ tar -xf l_openvino_toolkit_ie_p_2018.5.445.tgz
```

The result of untarring the archive is a folder called `inference_engine_vpu_arm` .

Step #5: Configure OpenVINO on your Raspberry Pi

Let' s modify the `setupvars.sh` script with the ***absolute path*** to our OpenVINO directory.

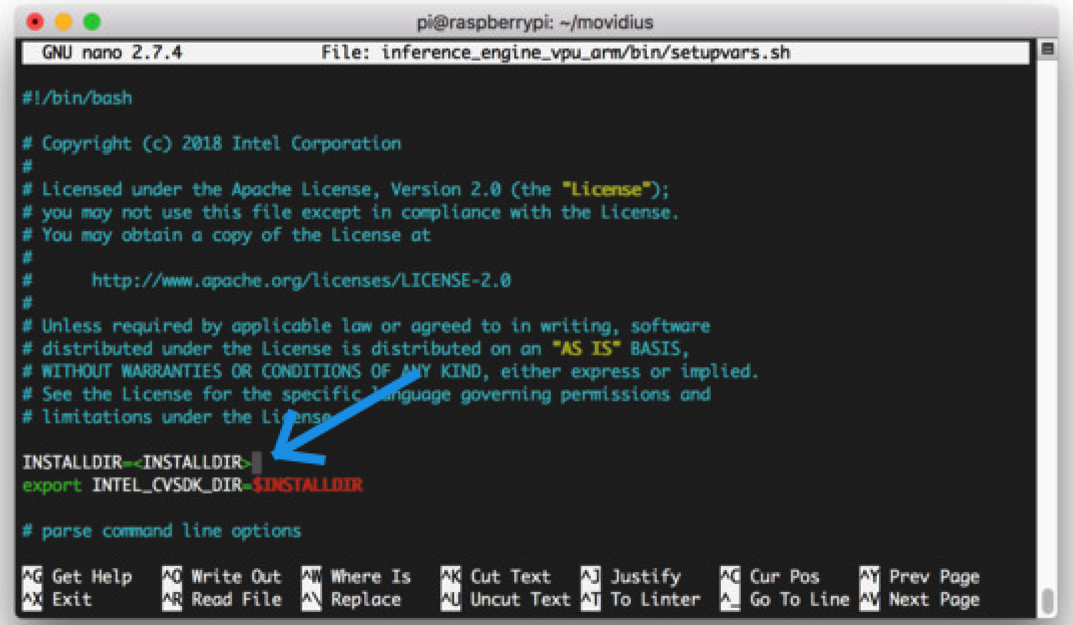
To do so, we' re going to use the nano terminal text file editor:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

```
$ nano openvino/inference_engine_vpu_arm/bin/setupvars.sh
```

The file will look like this:



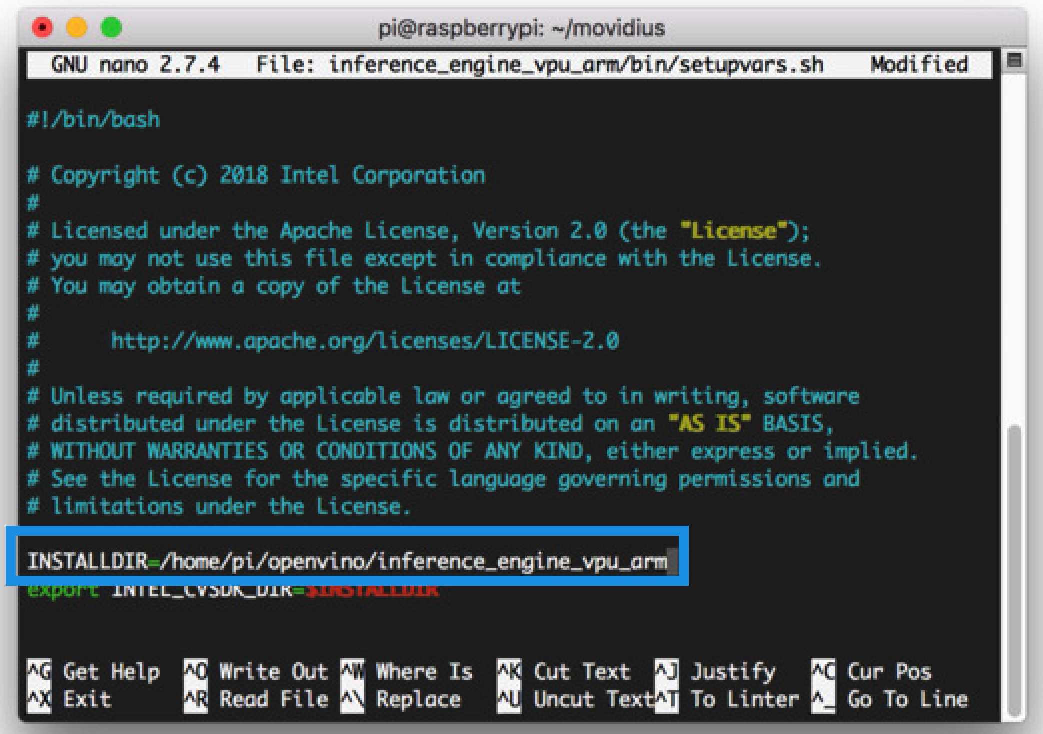
```
  GNU nano 2.7.4          File: inference_engine_vpu_arm/bin/setupvars.sh

#!/bin/bash

# Copyright (c) 2018 Intel Corporation
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#      http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

INSTALLDIR=<INSTALLDIR>
export INTEL_CVSDK_DIR=$INSTALLDIR

# parse command line options

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    ^Y Prev Page
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Linter  ^_ Go To Line ^V Next Page
```

Figure 5: Intel OpenVINO `setupvars.sh` file requires that you insert the path to the OpenVINO installation directory on the Raspberry Pi.

You need to replace `<INSTALLDIR>` with the following:

`/home/pi/openvino/inference_engine_vpu_arm`

It should now look just like so:



```
pi@raspberrypi: ~/movidius
  GNU nano 2.7.4    File: inference_engine_vpu_arm/bin/setupvars.sh     Modified

#!/bin/bash

# Copyright (c) 2018 Intel Corporation
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#      http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

INSTALLDIR=/home/pi/openvino/inference_engine_vpu_arm
export INTEL_CVSDK_DIR=$INSTALLDIR

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Linter ^_ Go To Line
```

Figure 8: The installation directory has been updated for OpenVINO' s `setupvars.sh` on the Raspberry Pi.

To save the file press *"ctrl + o, enter"* followed by *"ctrl + x "*to exit.

From there, let' s use `nano` again to edit our `~/.bashrc` . We will add a line to load OpenVINO' s `setupvars.sh` each time you invoke a Pi terminal. Go ahead and open the file:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

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

Scroll to the bottom and add the following lines:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

```
# OpenVINO
source ~/openvino/inference_engine_vpu_arm/bin/setupvars.sh
```

Now save and exit from nano as we did previously.

Then, go ahead and `source` your `~/.bashrc` file:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

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

Step #6: Configure USB rules for your Movidius NCS and OpenVINO on Raspberry Pi

OpenVINO requires that we set custom USB rules. It is quite straightforward, so let’ s get started.

First, enter the following command to add the current user to the Raspbian “users” group:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

```
$ sudo usermod -a -G users "$(whoami)"
```

Then logout and log back in. If you’ re on SSH, you can type `exit` and then re-establish your SSH connection. Rebooting is also an option via `sudo reboot now` .

Once you’ re back at your terminal, run the following script to set the USB rules:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

```
$ cd ~
$ sh openvino/inference_engine_vpu_arm/install_dependencies/install_NCS_udev_rules
```

Step #7: Create an OpenVINO virtual environment on Raspberry Pi

Let’ s grab and install pip, a Python Package Manager.

To install pip, simply enter the following in your terminal:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

```
$ wget https://bootstrap.pypa.io/get-pip.py
$ sudo python3 get-pip.py
```

We’ ll be making use of virtual environments for Python development with OpenCV and OpenVINO.

If you aren’ t familiar with virtual environments, please take a moment look at this article on RealPython or read the first half of this blog post on PyImageSearch.

Virtual environments will allow you to run independent, sequestered Python environments in isolation on your system. Today we’ ll be setting up just one environment, but you could easily have an environment for each project.

Let’ s go ahead and install `virtualenv` and `virtualenvwrapper` now — they allow for Python virtual environments:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

```
$ sudo pip install virtualenv virtualenvwrapper
$ sudo rm -rf ~/get-pip.py ~/.cache/pip
```

To finish the install of these tools, we need to update our `~/.bashrc` again:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

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

Then add the following lines:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

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

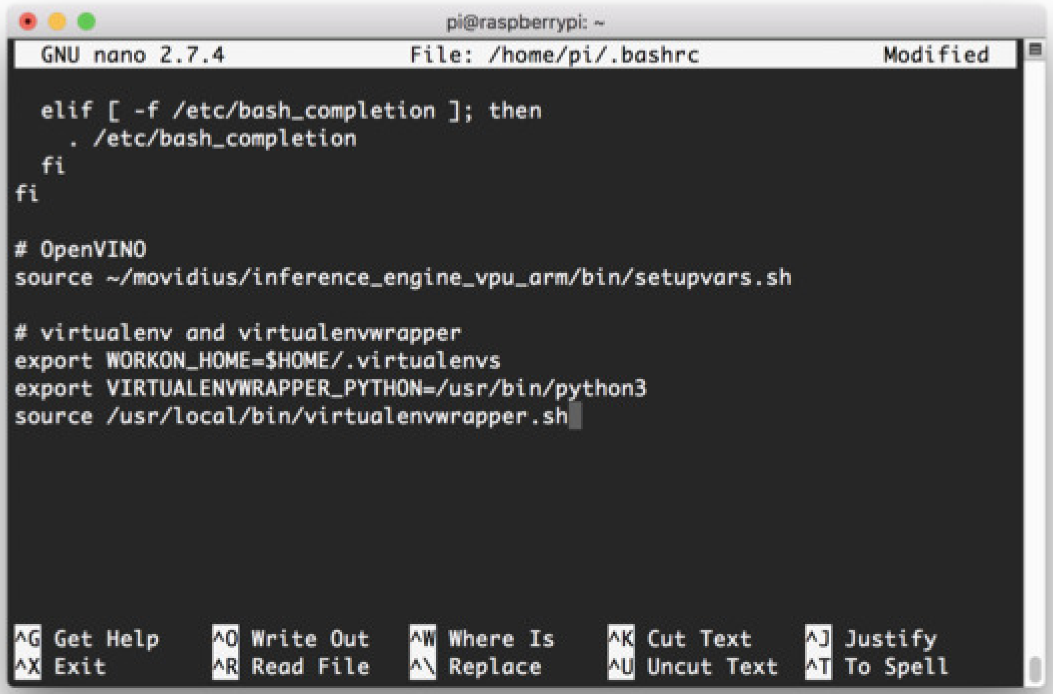

```
  GNU nano 2.7.4              File: /home/pi/.bashrc                Modified

    elif [ -f /etc/bash_completion ]; then
      . /etc/bash_completion
    fi
fi

# OpenVINO
source ~/movidius/inference_engine_vpu_arm/bin/setupvars.sh

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

Figure 9: Our Raspberry Pi ~/. bashrc profile has been updated to accommodate OpenVINO and virtualenvwrapper. Now we' ll be able to create a virtual environment for Python packages.

Alternatively, you can append the lines directly via bash commands:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

```
$ echo -e "\n# virtualenv and virtualenvwrapper" >> ~/.bashrc
$ echo "export WORKON_HOME=$HOME/.virtualenvs" >> ~/.bashrc
$ echo "export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3" >> ~/.bashrc
$ echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.bashrc
```

Next, source the `~/.bashrc` profile:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

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

Let' s now create a virtual environment to hold OpenVINO, OpenCV and related packages:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

```
$ mkvirtualenv openvino -p python3
```

This command simply creates a Python 3 virtual environment named `openvino` . You can (and should) name your environment(s) whatever you'd like — I like to keep them short and sweet while also providing enough information so I'll remember what they are for.

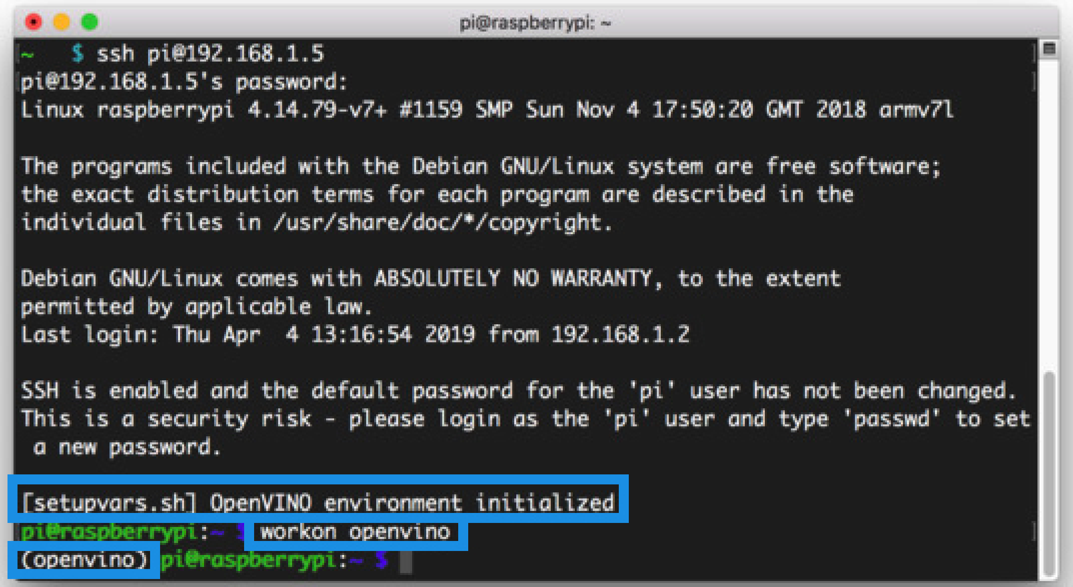
Let's verify that we're "in" the `openvino` environment by taking a look at the bash prompt. It should show `(openvino)` at the beginning of the prompt as shown in the image:

If your virtual environment is not active, you can simply use the `workon` command.:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

```
$ workon openvino
```



```
~ $ ssh pi@192.168.1.5
pi@192.168.1.5's password:
Linux raspberrypi 4.14.79-v7+ #1159 SMP Sun Nov 4 17:50:20 GMT 2018 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Apr  4 13:16:54 2019 from 192.168.1.2

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
 a new password.

[setupvars.sh] OpenVINO environment initialized
pi@raspberrypi:~ $ workon openvino
(openvino) pi@raspberrypi:~ $
```

Figure 10: The `workon openvino` command activates our OpenVINO Python 3 virtual environment. We're now ready to install Python packages and run computer vision code with Movidius and the Raspberry Pi.

Step #8: Install packages into your OpenVINO environment

Let's install a handful of packages required for today's demo script

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

```
$ workon openvino
$ pip install numpy
$ pip install "picamera[array]"
$ pip install imutils
```

Now that we’ ve installed these packages in the `openvino` virtual environment, they are only available in the `openvino` environment. This is your sequestered area to work on OpenVINO projects (we use Python virtual environments here so we don’ t risk ruining your system install of Python).

Additional packages for Caffe, TensorFlow, and mxnet may be installed via requirements.txt files using pip. You can read more about it at this Intel documentation link. This is ***not*** required for today’ s tutorial.

Step #6: Link OpenVINO’ s OpenCV into your Python 3 virtual environment

OpenCV is ready to go *outside* our virtual environment. But that’ s bad practice to use the system environment. Let’ s instead link the OpenVINO version of OpenCV into our Python virtual environment so we have it at our fingertips for today’ s demo (and whatever future projects you dream up).

Here we are going to create a “symbolic link” . A symbolic link creates a special linkage between two places on your system (in our case it is a `.so` file — think of a sym-link as a “shortcut” that points to another file.

When running the command you’ ll notice that we navigate into the destination of the link, and create our sym-link back to where the file actually lives.

I had a hard time finding OpenVINO’ s OpenCV `.so` file, so I used the `find` command:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

```
$ find / -name "cv2*.so"
...
/home/pi/openvino/inference_engine_vpu_arm/python/python3.5/cv2.cpython-35m-arm-linux-gnueabihf.so
```

I had to scroll through a bunch of output to find the OpenCV binary filepath. Thus, I’ ve omitted the unneeded output above.

Ensure that you copy the Python 3.5 path and not the Python 2.7 one since we’ re using Python 3.

From there, with the path in our clipboard, let’ s create our sym-link into the `openvino` virtual environment `site-packages` :

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Python

```
$ cd ~/.virtualenvs/openvino/lib/python3.5/site-packages/
$ ln -s /home/pi/openvino/inference_engine_vpu_arm/python/python3.5/cv2.cpython-35
arm-linux-gnueabihf.so cv2.so
$ cd ~
```

Take care to notice that the 2nd line wraps as it is especially long. I cannot stress this step enough — **this step is *critical.*** If you don’ t create a symbolic link, you won’ t be able to import OpenCV in your OpenVINO Python scripts. Also, ensure that the paths and filenames in the above commands are correct for your Raspberry Pi. **I suggest tab-completion.**

Step #7: Test your OpenVINO install on your Raspberry Pi

Let’ s do a quick sanity test to see if OpenCV is ready to go before we try an OpenVINO example.

Open a terminal and perform the following:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Python

```
$ workon openvino
$ python
>>> import cv2
>>> cv2.__version__
'4.0.1-openvino'
>>> exit()
```

The first command activates our OpenVINO virtual environment. From there we fire up the Python 3 binary in the environment and import OpenCV.

The version of OpenCV indicates that it is an OpenVINO optimized install!

Real-time object detection with Raspberry Pi and OpenVINO

Installing OpenVINO was pretty easy and didn’ t even require a compile of OpenCV. The Intel team did a great job!

Now let’ s put the Movidius Neural Compute Stick to work using OpenVINO.

For comparison’ s sake, we’ ll run the MobileNet SSD object detector *with* and *without* the Movidius to benchmark our FPS. We’ ll compare the values to previous results of using Movidius NCS APIv1 (the non-OpenVINO method that I wrote about in early 2018).

Let' s get started!

Project structure

Go ahead and grab the *"Downloads"* for today' s blog post.

Once you' ve extracted the zip, you can use the `tree` command to inspect the project directory:

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Shell

```
$ tree
.
├── MobileNetSSD_deploy.caffemodel
├── MobileNetSSD_deploy.prototxt
├── openvino_real_time_object_detection.py
└── real_time_object_detection.py

0 directories, 3 files
```

Our MobileNet SSD object detector files include the .caffemodel and .prototxt.txt files. These are pretrained (we will not be training MobileNet SSD today).

We' re going to review the `openvino_real_time_object_detection.py` script and compare it to the original real-time object detection script (`real_time_object_detection.py`).

Real-time object detection with OpenVINO, Movidius NCS, and Raspberry Pi

To demonstrate the power of OpenVINO on the Raspberry Pi with Movidius, we' re going to perform **real-time deep learning object detection.**

The Movidius/Myriad coprocessor will perform the actual deep learning inference, reducing the load on the Pi' s CPU.

We' ll still use the Raspberry Pi CPU to process the results and tell the Movidius what to do, but we' re reserving deep learning inference for the Myriad as its hardware is optimized and designed for deep learning inference.

As previously discussed in the *"What is OpenVINO?"* section, OpenVINO with OpenCV allows us to specify the processor for inference when using the OpenCV "DNN" module.

In fact, it only requires *one line of code* (typically) to use the Movidius NCS Myriad processor.
From there, the rest of the code is the same!
On the PyImageSearch blog I provide a detailed walkthrough of all Python scripts.
This is one of the few posts where I’ve decided to *deviate from my typical format.*
This post is first and foremost an *install + configuration* post. Therefore I’m going to skip over the details and instead demonstrate the power of OpenVINO by highlighting new lines of code inserted into a previous blog post (where all details are provided).
Please review that post if you want to get into the weeds with *Real-time object detection with deep learning and OpenCV* where I demonstrated the concept of using OpenCV’s DNN module in **just 100 lines of code.**
Today, we’re adding just one line of code that performs computation (and a comment + blank line). This brings the **new total** to ***103* lines of code** without using the previous complex Movidius APIv1 (**215 lines of code**).
If this is your first foray into OpenVINO, I think you’ll be just as astounded and pleased as I was when I learned how easy it is.
Let’s learn the changes necessary to accommodate OpenVINO’s API with OpenCV and Movidius.
Go ahead and open a file named `openvino_real_time_object_detection.py` and insert the following lines, paying close attention to **Lines 33-35** (highlighted in yellow):

Installing OpenVINO on the Raspberry Pi and performing object detection with the Movidius Neural Compute Stick

Python

```
1   # import the necessary packages
2   from imutils.video import VideoStream
3   from imutils.video import FPS
4   import numpy as np
5   import argparse
6   import imutils
7   import time
8   import cv2
9
10  # construct the argument parse and parse the arguments
11  ap = argparse.ArgumentParser()
12  ap.add_argument("-p", "--prototxt", required=True,
13          help="path to Caffe 'deploy' prototxt file")
```

```python
ap.add_argument("-m", "--model", required=True,
        help="path to Caffe pre-trained model")
ap.add_argument("-c", "--confidence", type=float, default=0.2,
        help="minimum probability to filter weak detections")
ap.add_argument("-u", "--movidius", type=bool, default=0,
        help="boolean indicating if the Movidius should be used")
args = vars(ap.parse_args())

# initialize the list of class labels MobileNet SSD was trained to
# detect, then generate a set of bounding box colors for each class
CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
        "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
        "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
        "sofa", "train", "tvmonitor"]
COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))

# load our serialized model from disk
print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])

# specify the target device as the Myriad processor on the NCS
net.setPreferableTarget(cv2.dnn.DNN_TARGET_MYRIAD)

# initialize the video stream, allow the cammera sensor to warmup,
# and initialize the FPS counter
print("[INFO] starting video stream...")
vs = VideoStream(usePiCamera=True).start()
time.sleep(2.0)
fps = FPS().start()

# loop over the frames from the video stream
while True:
        # grab the frame from the threaded video stream and resize it
        # to have a maximum width of 400 pixels
        frame = vs.read()
        frame = imutils.resize(frame, width=400)

        # grab the frame dimensions and convert it to a blob
        (h, w) = frame.shape[:2]
        blob = cv2.dnn.blobFromImage(frame, 0.007843, (300, 300), 127.5)

        # pass the blob through the network and obtain the detections and
        # predictions
        net.setInput(blob)
        detections = net.forward()

        # loop over the detections
        for i in np.arange(0, detections.shape[2]):
                # extract the confidence (i.e., probability) associated with
                # the prediction
                confidence = detections[0, 0, i, 2]

                # filter out weak detections by ensuring the `confidence` is
                # greater than the minimum confidence
                if confidence > args["confidence"]:
                        # extract the index of the class label from the
                        # `detections`, then compute the (x, y)-coordinates of
                        # the bounding box for the object
                        idx = int(detections[0, 0, i, 1])
                        box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
                        (startX, startY, endX, endY) = box.astype("int")

                        # draw the prediction on the frame
                        label = "{}: {:.2f}%".format(CLASSES[idx],
                                confidence * 100)
                        cv2.rectangle(frame, (startX, startY), (endX, endY),
                                COLORS[idx], 2)
                        y = startY - 15 if startY - 15 > 15 else startY + 15
                        cv2.putText(frame, label, (startX, y),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)

        # show the output frame
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF

        # if the `q` key was pressed, break from the loop
        if key == ord("q"):
                break

        # update the FPS counter
        fps.update()

# stop the timer and display FPS information
fps.stop()
print("[INFO] elasped time: {:.2f}".format(fps.elapsed()))
print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))

# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```

Lines 33-35 (highlighted in yellow) are new. But only one of those lines is interesting.

On **Line 35**, we tell OpenCV’ s DNN module to use the Myriad coprocessor using `net.setPreferableTarget(cv2.dnn.DNN_TARGET_MYRIAD)` .

The Myriad processor is built into the Movidius Neural Compute Stick. You can use this same method if you’ re running OpenVINO + OpenCV on a device with an embedded Myriad chip (i.e. without the bulky USB stick).

For a detailed explanation on the code, be sure to refer to this post.

Also, be sure to refer to this Movidius APIv1 blog post from early 2018 where I demonstrated object detection using Movidius and the Raspberry Pi. It’ s incredible that 215 lines of significantly more complicated code are required for the previous Movidius API, in comparison to 103 lines of much easier to follow code using OpenVINO.

I think those line number differences speak for themselves in terms of reduced complexity, time, and development cost savings, ***but what are the actual results?*** **How fast is OpenVINO with Movidius?**

Let’ s find out in the next section.

OpenVINO object detection results

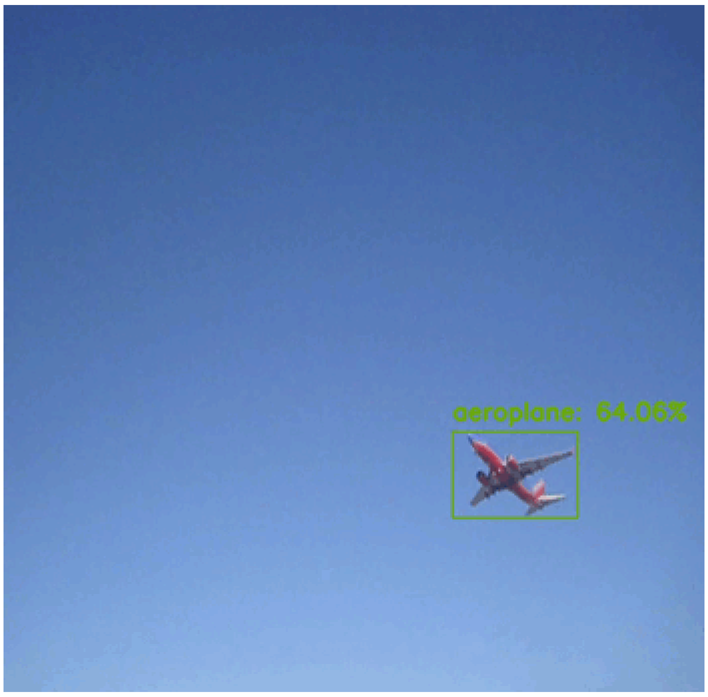


FIgure **11:** Object detection with OpenVINO, OpenCV, and the Raspberry Pi.

To run today’ s script, first, you’ ll need to grab the ***“Downloads”*** associated with this post.

From there, unpack the zip and navigate into the directory.

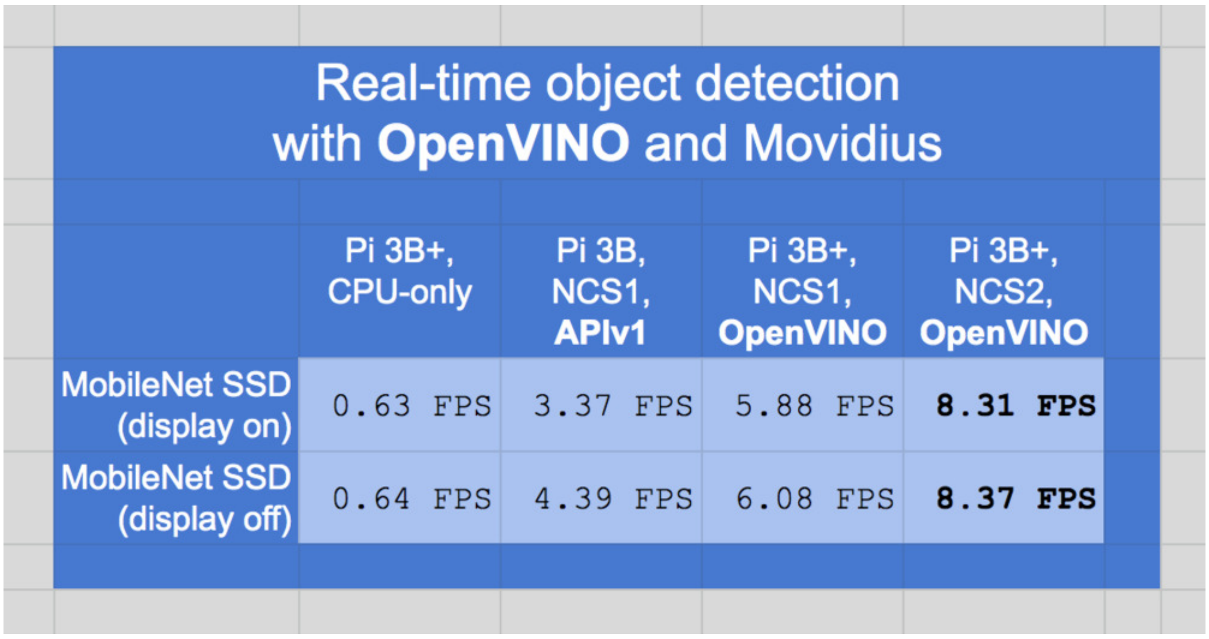
To perform object detection with OpenVINO, just execute the following command:

```
Installing OpenVINO on the Raspberry Pi and performing object detection with the
Movidius Neural Compute Stick
Shell
```

```
$ python openvino_real_time_object_detection.py
        --prototxt MobileNetSSD_deploy.prototxt \
        --model MobileNetSSD_deploy.caffemodel
[INFO] loading model...
[INFO] starting video stream...
[INFO] elasped time: 55.35
[INFO] approx. FPS: 8.31
```

As you can see, we’ re reaching **8.31FPS** over approximately one minute.

I’ ve gathered additional results using MobileNet SSD as shown in the table below:



Real-time object detection with **OpenVINO** and Movidius

	Pi 3B+, CPU-only	Pi 3B, NCS1, **APIv1**	Pi 3B+, NCS1, **OpenVINO**	Pi 3B+, NCS2, **OpenVINO**
MobileNet SSD (display on)	0.63 FPS	3.37 FPS	5.88 FPS	**8.31 FPS**
MobileNet SSD (display off)	0.64 FPS	4.39 FPS	6.08 FPS	**8.37 FPS**

Figure 12: A benchmark comparison of the MobileNet SSD deep learning object detector using OpenVINO with the Movidius Neural Compute Stick.

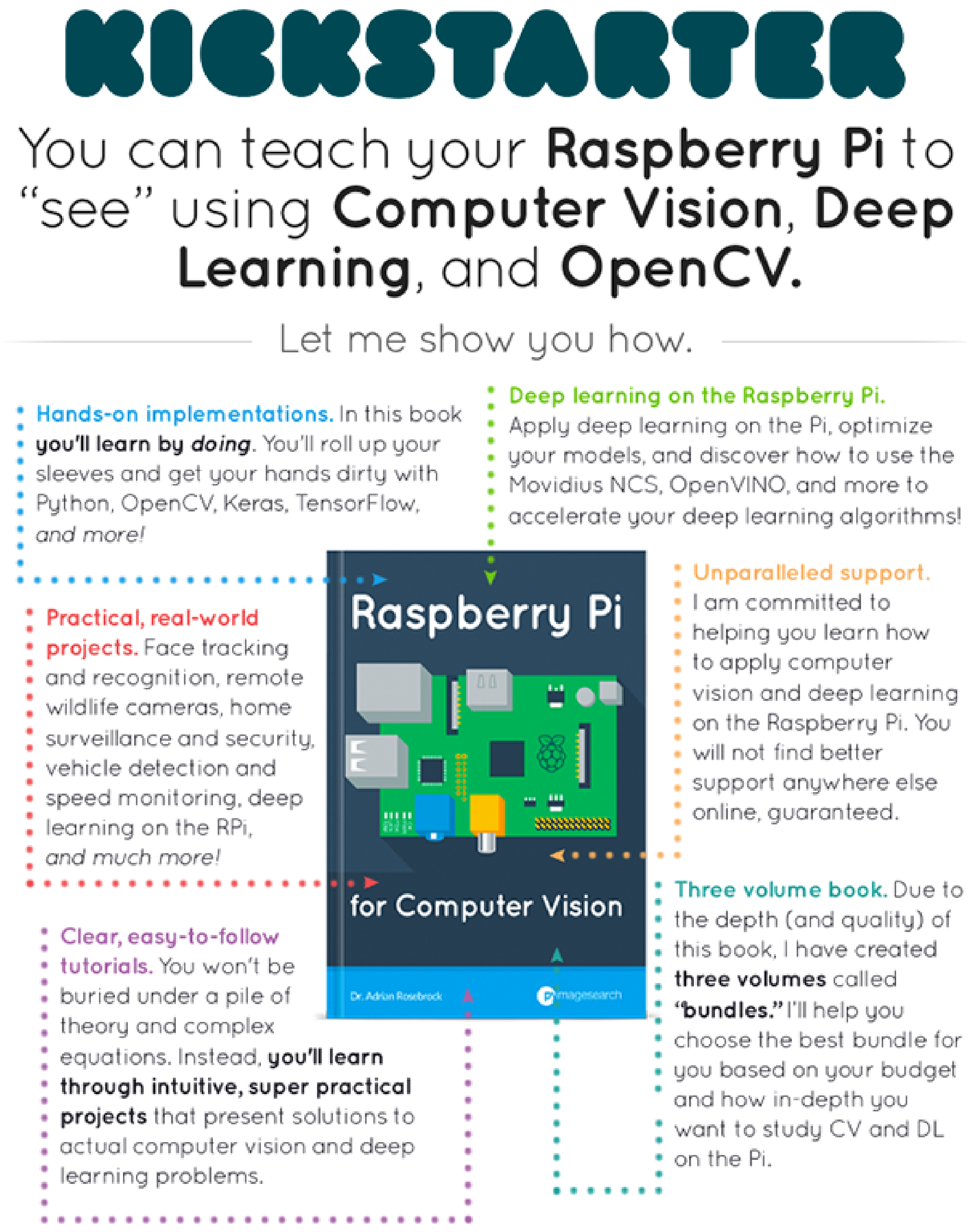
OpenVINO and the Movidius NCS 2 are *very fast*, a *huge* speedup from previous versions.

It’ s amazing that the results are > 8x in comparison to using only the RPi 3B+ CPU (no Movidius coprocessor).

The two rightmost columns (light blue columns 3 and 4) show the OpenVINO comparison between the NCS1 and the NCS2.

Note that the 2nd column statistic is with the RPi 3B (not the 3B+). It was taken in February 2018 using the previous API and previous RPi hardware.

So, what’ s next?


KICKSTARTER

You can teach your **Raspberry Pi** to "see" using **Computer Vision**, **Deep Learning**, and **OpenCV.**

Let me show you how.

Hands-on implementations. In this book **you'll learn by *doing***. You'll roll up your sleeves and get your hands dirty with Python, OpenCV, Keras, TensorFlow, *and more!*

Deep learning on the Raspberry Pi. Apply deep learning on the Pi, optimize your models, and discover how to use the Movidius NCS, OpenVINO, and more to accelerate your deep learning algorithms!

Practical, real-world projects. Face tracking and recognition, remote wildlife cameras, home surveillance and security, vehicle detection and speed monitoring, deep learning on the RPi, *and much more!*

Unparalleled support. I am committed to helping you learn how to apply computer vision and deep learning on the Raspberry Pi. You will not find better support anywhere else online, guaranteed.

Clear, easy-to-follow tutorials. You won't be buried under a pile of theory and complex equations. Instead, **you'll learn through intuitive, super practical projects** that present solutions to actual computer vision and deep learning problems.

Three volume book. Due to the depth (and quality) of this book, I have created **three volumes** called **"bundles."** I'll help you choose the best bundle for you based on your budget and how in-depth you want to study CV and DL on the Pi.

Figure 13: The *Raspberry Pi for Computer Vision* book Kickstarter begins on Wednesday, April 10, 2019 at 10am EDT.

I' m currently getting code and materials together to start writing a new ***Raspberry Pi for Computer Vision*** **book.**

The book will cover everything needed to maximize computer vision + deep learning capability on resource-constrained devices such as the Raspberry Pi single board computer (SBC).

You' ll learn and develop your skills using techniques that I' ve amassed through my years of working with computer vision on the Raspberry Pi and other devices.

The book will come with **over 40 chapters** with **tons of working code.**

Included are **preconfigured Raspbian .img files** (for the Raspberry Pi 3B+/3B and Raspberry Pi Zero W) so you can skip the tedious installation headaches and get to the fun part (code and deployment).

Sound interesting?

- You can read the **official book announcement here.**
- I’ve also put together a **sneak preview video** so that you know just what to expect in my upcoming book. You don’t want to miss this video!
- My **tentative chapter listing/table of contents is available here** as well.
- Finally, be sure to **mark your calendar for Wednesday at 10AM EDT** when the **Kickstarter goes live!**

So what do you say?

Are you interested in learning how to use the Raspberry Pi for computer vision and deep learning?

If so, be sure to click the button below and enter your email address to receive book updates to your inbox:

Keep me in the loop!

Troubleshooting and Frequently Asked Questions (FAQ)

Did you encounter an error installing OpenCV and OpenVINO on your Raspberry Pi?

Don’t throw the blue USB stick into the toilet just yet.

The first time you install the software on your Raspberry Pi it can be very frustrating. The last thing I want for you to do is give up!

Here are some common question and answers — be sure to read them and see if they apply to you.

Q. How do I flash an operating system on to my Raspberry Pi memory card?

A. I recommend that you:

- Grab a 16GB or 32GB memory card.
- Flash Raspbian Stretch with Etcher to the card. Etcher is supported by most major operating systems.

- Insert the card into your Raspberry Pi and begin with the *"Assumptions"* and *"Step 1" sections* in this blog post.

Q. Can I use Python 2.7?

A. I don't recommend using Python 2.7 as it's rapidly approaching its end of life. Python 3 is the standard now. I also haven't tested OpenVINO with Python 2.7. But if you insist...

Here's how to get up and running with Python 2.7:

How to install OpenCV 4 on Ubuntu

Shell

```
$ sudo apt-get install python2.7 python2.7-dev
```

Then, before you create your virtual environment in **Step #4**, first install pip for Python 2.7:

How to install OpenCV 4 on Ubuntu

Shell

```
$ sudo python2.7 get-pip.py
```

Also in **Step #4**: when you create your virtual environment, simply use the relevant Python version flag:

How to install OpenCV 4 on Ubuntu

Python

```
$ mkvirtualenv openvino_py27 -p python2.7
```

From there everything should be the same.

Q. Why can't I just apt-get install OpenCV and have OpenVINO support?

A. Avoid this "solution" *at all costs* even though it *might* work. First, this method likely won't install OpenVINO until it is more popular. Secondly, apt-get doesn't play nice with virtual environments and you won't have control over your compile and build.

Q. The `mkvirtualenv` and `workon` commands yield a "command not found error". I'm not sure what to do next.

A. There a number of reasons why you would be seeing this error message, all of come from to **Step #4:**

1. First, ensure you have installed `virtualenv` and `virtualenvwrapper` properly using the `pip` package manager. Verify by running `pip freeze`

and ensure that you see both `virtualenv` and `virtualenvwrapper` are in the list of installed packages.

2. Your `~/.bashrc` file may have mistakes. Examine the contents of your `~/.bashrc` file to see the proper `export` and `source` commands are present (check **Step #4** for the commands that should be appended to `~/.bashrc`).
3. You might have forgotten to `source` your `~/.bashrc` . Make sure you run `source ~/.bashrc` after editing it to ensure you have access to the `mkvirtualenv` and `workon` commands.

Q. When I open a new terminal, logout, or reboot my Raspberry Pi, I cannot execute the `mkvirtualenv` or `workon` commands.

A. If you' re on the Raspbian desktop, this will likely occur. The default profile that is loaded when you launch a terminal, for some reason, doesn' t source the `~/.bashrc` file. Please refer to **#2** from the previous question. Over SSH, you probably won' t run into this.

Q. When I try to import OpenCV, I encounter this message: `Import Error: No module named cv2` .

A. There are *several* reasons this could be happening and unfortunately, it is hard to diagnose. I recommend the following suggestions to help diagnose and resolve the error:

1. Ensure your `openvino` virtual environment is active by using the `workon openvino` command. If this command gives you an error, then verify that `virtualenv` and `virtualenvwrapper` are properly installed.
2. Try investigating the contents of the `site-packages` directory in your `openvino` virtual environment. You can find the `site-packages` directory in `~/.virtualenvs/openvino/lib/python3.5/site-packages/` . Ensure (1) there is a `cv2` sym-link directory in the `site-packages` directory and (2) it' s properly sym-linked.
3. Be sure to `find` the `cv2*.so` file as demonstrated in **Step #6**.

***Q.* What if my question isn' t listed here?**

A. Please leave a comment below or [send me an email](). If you post a comment below, just be aware that *code doesn' t format well in the* *comment form and I may have to respond to you via email instead.*

Summary

Today we learned about Intel' s OpenVINO toolkit and how it can be used to improve deep learning inference speed on the Raspberry Pi.

You also learned how to install the OpenVINO toolkit, including the OpenVINO-optimized version of OpenCV on the Raspberry Pi.

We then ran a simple MobileNet SSD deep learning object detection model. **It only required one line of code** to set the target device to the Myriad processor on the Movidius stick.

We also demonstrated that the Movidius NCS + OpenVINO is quite fast, *dramatically* outperforming object detection speed on the Raspberry Pi' s CPU.

And if you' re interested in learning more about how to build real-world computer vision + deep learning projects on the Raspberry Pi, be sure to check out my upcoming book, *Raspberry Pi for Computer Vision.* **I' ll be launching a Kickstarter pre-sale which begins just two days from now on *Wednesday, April 10th at 10AM EDT.***

Mark your calendar to take advantage of pre-sale bargain prices on the RPi book — see you then!