* The BLUE highlights are the original commands
* The YELLOW were found online and are new
* The non-highlighted are also original but were not used
* I also installed OpenCV before installing tensorflow
* I’m using legacy 32bit full (inside “raspberry pi OS (other)”)

A screenshot of a computer

Description automatically generated

---------------------------------------------------------------------

SETTING UP ENVIRONMENT FOR FINAL PROJECT

---------------------------------------------------------------------

1. First step, update system.

$ sudo apt-get update

$ sudo apt-get upgrade

sudo apt-get install python3-pip python3-virtualenv

---------------------------------------------------------------------

INSTALLING PYENV (so we can select which python version)

---------------------------------------------------------------------

2. Install pyenv.

$ curl https://pyenv.run | bash

3. Add pyenv to .bashrc script

$ sudo nano ~/.bashrc

Add the following lines at the end of the file

export PATH="$HOME/.pyenv/bin:$PATH"

eval "$(pyenv init --path)"

eval "$(pyenv virtualenv-init -)"

export PYENV\_ROOT="$HOME/.pyenv"

[[ -d $PYENV\_ROOT/bin ]] && export PATH="$PYENV\_ROOT/bin:$PATH"

eval "$(pyenv init -)"

eval "$(pyenv virtualenv-init -)"

Save and exit

5. Restart terminal to apply changes.

$ exec $SHELL

Or just close terminal and start a new one, or exit ssh and log

back in, etc.

6. make sure it worked

$ pyenv --version

You should get an output. If it says the command is not recognized,

something went wrong.

7. Install system packages.

$ sudo apt-get install --yes libatlas-base-dev libjasper-dev libilmbase-dev libopenexr-dev libgstreamer1.0-dev libssl-dev zlib1g-dev libbz2-dev libreadline-dev libsqlite3-dev llvm libncurses5-dev libncursesw5-dev xz-utils tk-dev libgdbm-dev lzma lzma-dev tcl-dev libxml2-dev libxmlsec1-dev libffi-dev liblzma-dev wget curl make build-essential openssl

8. Update pyenv.

$ pyenv update

9. Install Python version 3.7.12.

$ pyenv install --list

to show available versions, we want 3.7.12 so we will type

$ pyenv install 3.7.12

10. Check python version.

$ python --version

will probably be 3.9.x, 3.10.x, etc.

11. Make and enter a new folder for your project.

$ mkdir project

$ cd project

12. Use pyenv to change the python version within the folder.

$ pyenv local 3.7.12

Check python version again, it should now say 3.7.12

If you exit the folder and check the python version,

it should still be whatever it was originally.

If you want to change the version everywhere, you can use

$ pyenv global <version>

---------------------------------------------------------------------

CREATING VIRTUAL ENVIRONMENT WITH VENV

---------------------------------------------------------------------

13. Create a virtual environment (inside project folder).

$ python -m venv env

(optionally, 'env' can be replaced with a name of your choice)

14. Activate the virtual environment.

$ source env/bin/activate

if you want to exit the virtual environment

$ deactivate

$ sudo apt install -y build-essential cmake pkg-config libjpeg-dev libtiff5-dev libpng-dev libavcodec-dev libavformat-dev libswscale-dev libv4l-dev libxvidcore-dev libx264-dev libfontconfig1-dev libcairo2-dev libgdk-pixbuf2.0-dev libpango1.0-dev libgtk2.0-dev libgtk-3-dev libatlas-base-dev gfortran libhdf5-dev libhdf5-serial-dev libhdf5-103 libqt5gui5 libqt5webkit5 libqt5test5 python3-pyqt5 python3-dev

$ pip install "picamera[array]"

----------------------------------------------------------------------

INSTALL PACKAGES IN VIRTUAL ENVIRONMENT

----------------------------------------------------------------------

15. Update pip.

$ python -m pip install --upgrade pip

Not always necessary, but it doesn't hurt.

16. Install more system packages.

$ sudo apt-get install -y libhdf5-dev libc-ares-dev libeigen3-dev gcc gfortran libgfortran5 libatlas3-base libatlas-base-dev libopenblas-dev libopenblas-base libblas-dev liblapack-dev cython3 libatlas-base-dev openmpi-bin libopenmpi-dev python3-dev build-essential cmake pkg-config libjpeg-dev libtiff5-dev libpng-dev libavcodec-dev libavformat-dev libswscale-dev libv4l-dev libxvidcore-dev libx264-dev libfontconfig1-dev libcairo2-dev libgdk-pixbuf2.0-dev libpango1.0-dev libgtk2.0-dev libgtk-3-dev libhdf5-serial-dev libhdf5-103 libqt5gui5 libqt5webkit5 libqt5test5 python3-pyqt5 libatlas-base-dev libjasper-dev libilmbase-dev libopenexr-dev libgstreamer1.0-dev

$ sudo apt-get install -y libxrandr2 libdatrie1 libtheora0 libxcb-shm0 libavutil56 libopus0 libxfixes3 libcairo2 libswscale5 libspeex1 libaom0 libvorbisenc2 libwayland-egl1 libva2 libtiff5 libbluray2 libx265-165 libxvidcore4 libzvbi0 libsoxr0 libcroco3 libvorbis0a libavformat58 libcairo-gobject2 libmpg123-0 libogg0 libxrender1 libwayland-cursor0 libswresample3 libdrm2 libopenmpt0 libgme0 libvorbisfile3 libwebp6 libsnappy1v5 libtwolame0 libjbig0 libchromaprint1 libpango-1.0-0 libpixman-1-0 libatspi2.0-0 libvdpau1 libssh-gcrypt-4 libva-x11-2 libgsm1 libxinerama1 libx264-155 libwavpack1 libcodec2-0.8.1 libharfbuzz0b libepoxy0 libatlas3-base libxkbcommon0 libgfortran5 libxcb-render0 libfontconfig1 libpangoft2-1.0-0 libgdk-pixbuf2.0-0 libwayland-client0 libpangocairo-1.0-0 libxcomposite1 libvpx5 libxdamage1 libgtk-3-0 libva-drm2 libgraphite2-3 libavcodec58 libatk-bridge2.0-0 libopenjp2-7 librsvg2-2 libshine3 libxi6 libatk1.0-0 libwebpmux3 libthai0 libmp3lame0 libxcursor1

This is a lot. Some may be redundant, but might as well be safe.

19. Install OpenCV.

$ wget https://www.piwheels.org/simple/opencv-python/opencv\_python-4.7.0.72-cp37-cp37m-linux\_armv7l.whl#sha256=70ee61195da18e4ee2e812ef7da848172915c04aa25251723942dc6ca584fffc

pip install --upgrade pip setuptools

choose 1 option (3rd is best, once you choose 1 stick to it if you have to unistall and re-install)

$ pip install opencv-python -> let it autocomplete

$ pip install --no-cache-dir opencv-python

$ pip install opencv\_python-4.7.0.72-cp37-cp37m-linux\_armv7l.whl

17. If you're running Python 3.7.x on 32-bit OS (most people)

$ wget https://raw.githubusercontent.com/PINTO0309/Tensorflow-bin/main/previous\_versions/download\_tensorflow-2.4.0-cp37-none-linux\_armv7l.sh

Otherwise, will need to go here: https://github.com/PINTO0309/Tensorflow-bin and find shell script version matching Python version and Arch, then select, click view Raw, then copy url and do the above command with that url (and in the following commands in next step)

18. Install tensorflow.

$ chmod +x download\_tensorflow-2.4.0-cp37-none-linux\_armv7l.sh

$ ./download\_tensorflow-2.4.0-cp37-none-linux\_armv7l.sh

Remember you can press tab to autocomplete filename

$ sudo chmod +x https://raw.githubusercontent.com/PINTO0309/Tensorflow-bin/main/previous\_versions/download\_tensorflow-2.4.0-cp37-none-linux\_armv7l.sh

$ ./https://raw.githubusercontent.com/PINTO0309/Tensorflow-bin/main/previous\_versions/download\_tensorflow-2.4.0-cp37-none-linux\_armv7l.sh

$ pip install tensorflow-2.4.0-cp37-none-linux\_armv7l.whl

20. Check things are working.

Restart shell

$ exec $SHELL

Reactivate virtual environment

$ source env/bin/activate

Open python intepreter

$ python

>> import tensorflow as tf

>> tf.\_\_version\_\_

>> import numpy as np

>> np.\_\_version\_\_

>> import cv2

>> cv2.\_\_version\_\_

>> exit()

Might get a warning about h5py when importing tensorflow, ignore

that for now.

Note: other packages you will or might need in virtual environment

sense\_hat

skimage (pip install scikit-image)