Object Recognition Tutorial Using PYNQ Z2 Board

A detailed guide for object detection using deep learning approach on PYNQ Z2 board

Sarala Kumari Surapally
Id: 1952794
December 2020

INTRODUCTION

Object Detection and Recognition is one of the most demanding and challenging jobs in various applications such as autonomous vehicles, Crowd Counting, Face recognition etc. Object Detection involves classifying the objects in an image and localizing them by drawing bounding boxes around the objects in an image.

Object Detection can be done using Machine Learning based techniques and Deep Learning based techniques. In the Machine learning-based approach, the main focus is on various features of an image like color histogram or edges in order to identify the group of pixels that may belong to an object. These features are then passed to a model to predict object location and its label. However, Convolutional Neural Networks (CNNs) are used to achieve object detection in the case of Deep Learning-based approaches.

PYNQ is Python Productivity for Zynq. It is an open-source project from Xilinx that facilitates the design of systems integrated with Zynq All Programmable Systems on Chips (APSoCs). PYNQ-Z2 is an FPGA-based development platform that belongs to the ZYNQ XC7Z020 FPGA family, specifically designed to support PYNQ. Taking advantage of ARM programming and sophisticated software at ZYNQ, designers can build more powerful systems with it. Additionally, SoCs can be processed in Python using jupyter notebook and the code can be developed and implemented directly in PYNQ-Z2. Programmable programs are introduced as tool libraries and program libraries are imported and programmed in the same way as programmed APIs.

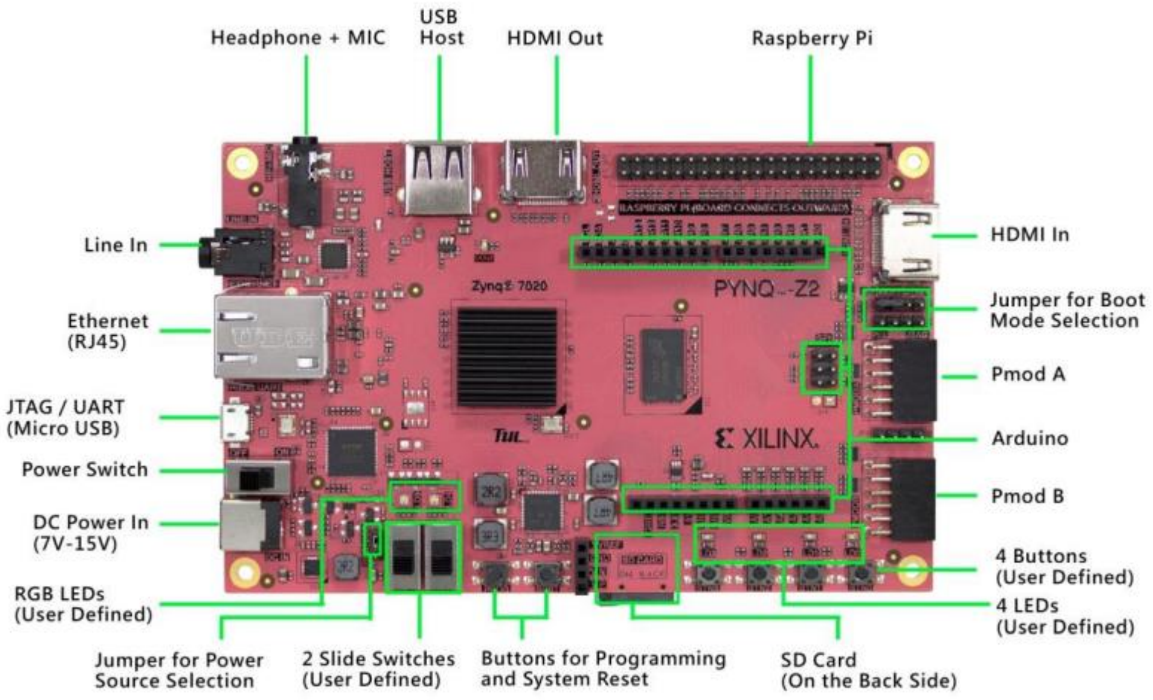
In this tutorial, we implement the object detection on the PYNQ Z2 board. This tutorial explains the basic concepts of the PYNQ Z2 board and some of the open-source examples of object recognition available on pynq website. The report is prepared with an assumption that the reader has knowledge of the basics of jupyter notebook and no prior knowledge on the pynq board.

To continue with the project, we need to have a PYNQ Z2 board, SD card preloaded with pynq image, Ethernet cable, USB cable, and web camera to capture images at run time.

PYNQ Z2 BOARD FEATURES:

PYNQ-Z2 board integrates USB and Ethernet to connect to internet, HDMI Input/Output, MIC Input, Audio Output, Arduino interface, Raspberry Pi interface, 2 Pmod, user LED, push-buttons, switches, MicroSD Slot, Power In port for direct power connection, a Jumper for power source selection and another jumper to select Boot Mode.

It is intended to be easily extensible with Pmod, Arduino, and peripherals, along with general-purpose GPIO pins.



MicroSD CARD SETUP:

You can download the pre-compiled PYNQ-Z2 image using the link http://www.pynq.io/board.html. Connect the microSD card to a PC/Laptop. Open Win32Disk Imager application, select the downloaded pynq image, and click on the write button. This will transfer the pynq image to SD card.

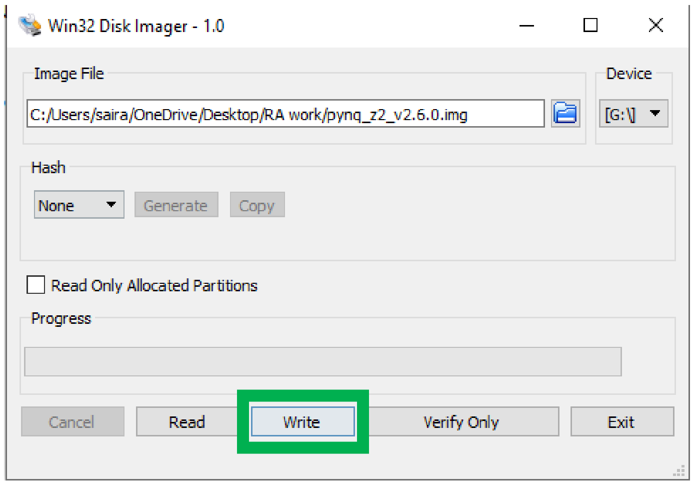

Win32 Disk Imager - 1.0

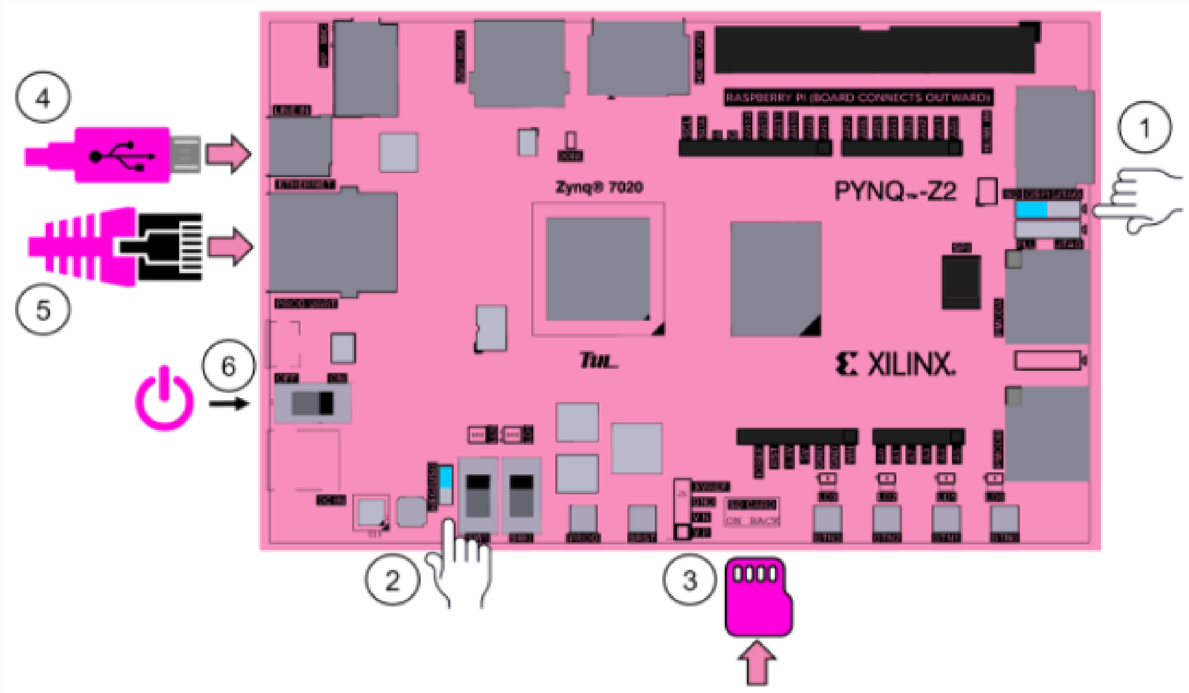
Image File: C:/Users/saira/OneDrive/Desktop/RA work/pynq_z2_v2.6.0.img

Device: [G:\]

Hash: None

BOARD SETUP:

Setup the board by following below steps.

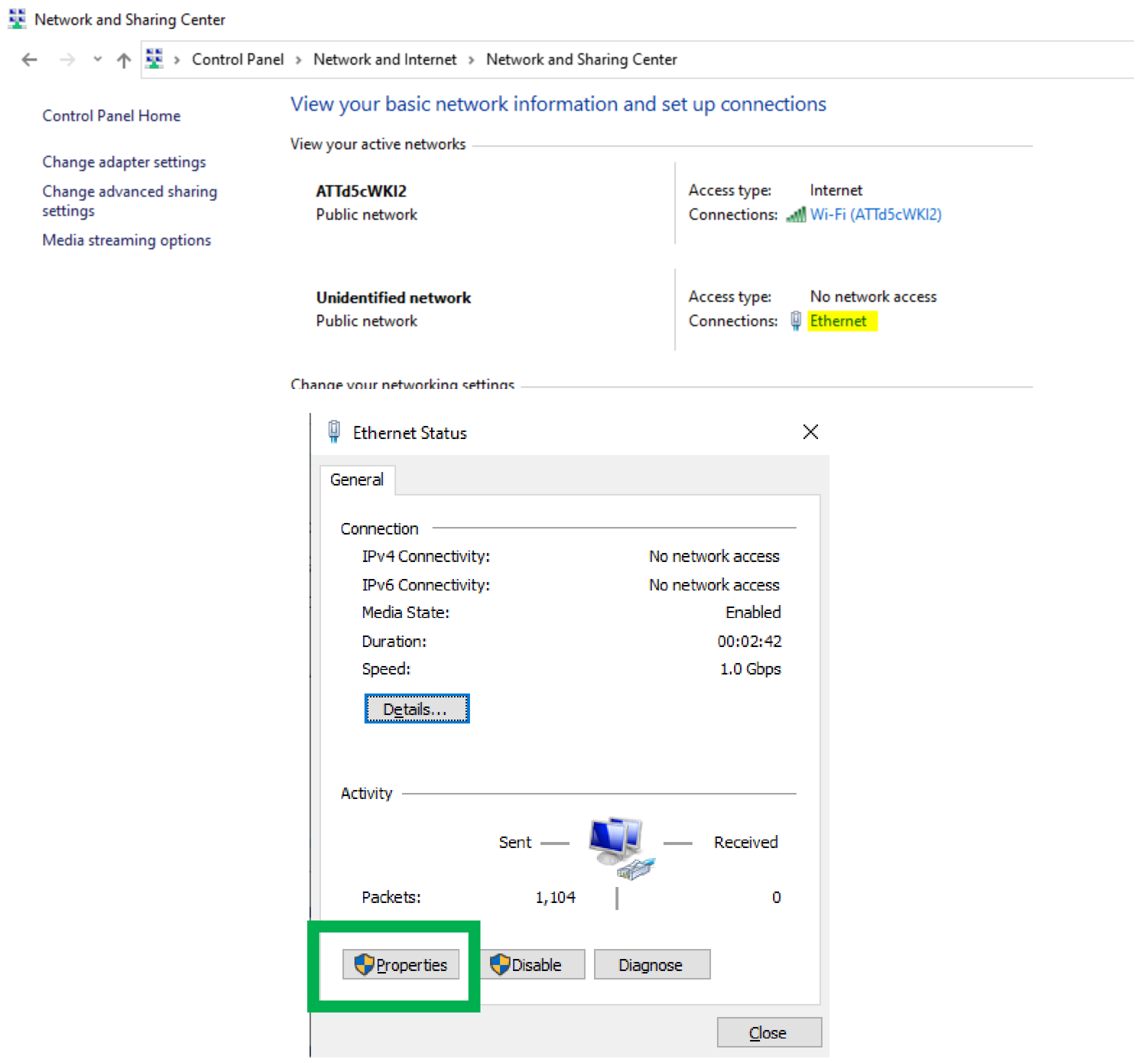


1. Set the boot jumper to SD card.
2. Set the power jumper to power the board from USB.
3. Insert the SD card loaded with pynq image.
4. Connect MicroUSB cable to PROG-UART port of the board and USB port to your PC/Laptop.
5. Connect Ethernet cable to a PC or to a Router.
6. Turn ON the board which will immediately glow the Red LED to indicate that the board has power and then the Done LED will turn ON to confirm that the device is working. Finally, 2 Blue LEDs and four Green LEDs will light up to show that the device is booted and ready to use.

NETWORK CONNECTION:

Set the IP address of your PC in the same range as the board by following the below steps:

- Open Control Panel -> Select Network and Internet -> Network and Sharing Center and then click on Ethernet (highlighted below)



Network and Sharing Center

Control Panel > Network and Internet > Network and Sharing Center

Control Panel Home
Change adapter settings
Change advanced sharing settings
Media streaming options

View your basic network information and set up connections

View your active networks

ATTd5cWKI2
Public network
Access type: Internet
Connections: Wi-Fi (ATTd5cWKI2)

Unidentified network
Public network
Access type: No network access
Connections: Ethernet

Change your networking settings

Ethernet Status

General

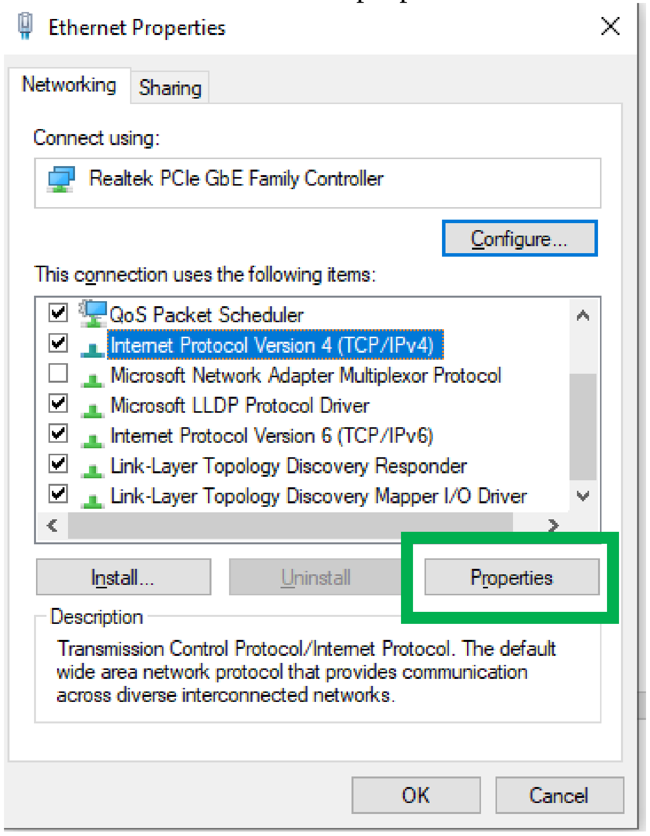
Connection

IPv4 Connectivity:	No network access
IPv6 Connectivity:	No network access
Media State:	Enabled
Duration:	00:02:42
Speed:	1.0 Gbps

Activity

	Sent	Received
Packets:	1,104	0

- Select Internet Protocol Version 4 and click on properties.



Ethernet Properties

Networking | Sharing

Connect using:

Realtek PCIe GbE Family Controller

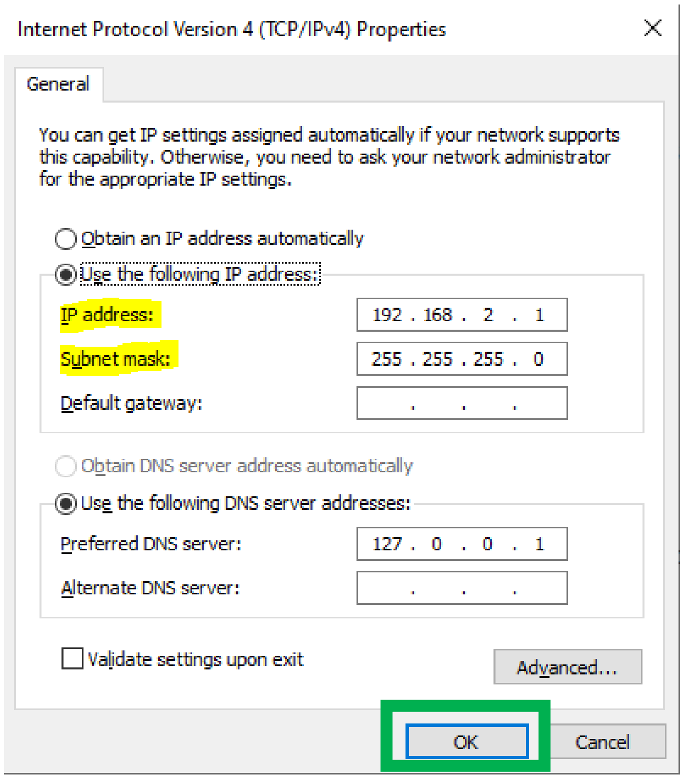
This connection uses the following items:

- [✓] QoS Packet Scheduler
- [✓] Internet Protocol Version 4 (TCP/IPv4)
- [ ] Microsoft Network Adapter Multiplexor Protocol
- [✓] Microsoft LLDP Protocol Driver
- [✓] Internet Protocol Version 6 (TCP/IPv6)
- [✓] Link-Layer Topology Discovery Responder
- [✓] Link-Layer Topology Discovery Mapper I/O Driver

Description

Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

- Set the IP Address to 192.168.2.1 and Subnet mask to 255.255.255.0 and click ok



Internet Protocol Version 4 (TCP/IPv4) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

○ Obtain an IP address automatically
● Use the following IP address:

IP address: 192 . 168 . 2 . 1
Subnet mask: 255 . 255 . 255 . 0
Default gateway: . . .

○ Obtain DNS server address automatically
● Use the following DNS server addresses:

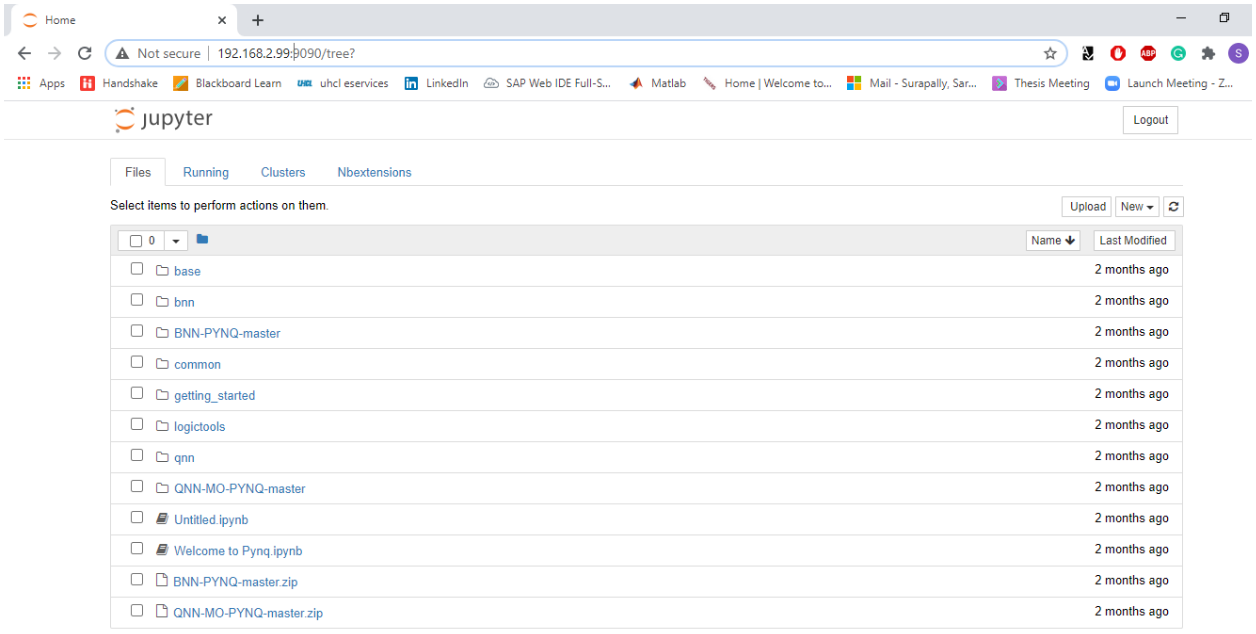
Preferred DNS server: 127 . 0 . 0 . 1
Alternate DNS server: . . .

[ ] Validate settings upon exit

CONNECTING TO JUPYTER NOTEBOOK:

Open a web browser and navigate to **http://192.168.2.99** to connect to jupyter notebook. You will be navigated to a login screen if the board setup is completed successfully. Enter Username xilinx and password xilinx.

Jupyter Notebook opens as below:

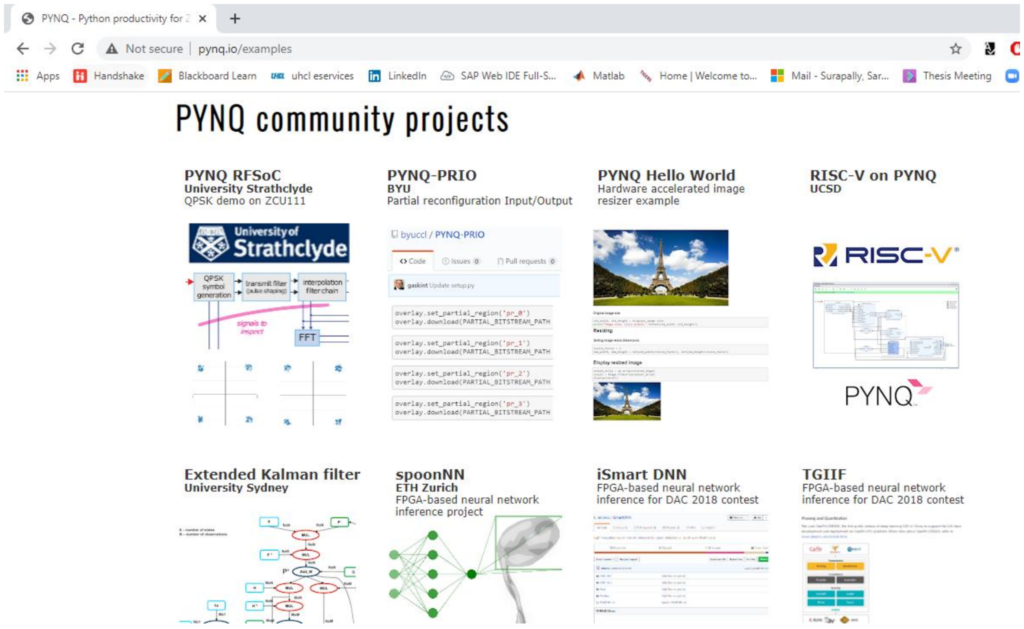


Select items to perform actions on them.

Name	Last Modified
base	2 months ago
bnn	2 months ago
BNN-PYNQ-master	2 months ago
common	2 months ago
getting_started	2 months ago
logictools	2 months ago
qnn	2 months ago
QNN-MO-PYNQ-master	2 months ago
Untitled.ipynb	2 months ago
Welcome to Pynq.ipynb	2 months ago
BNN-PYNQ-master.zip	2 months ago
QNN-MO-PYNQ-master.zip	2 months ago

INSTALLING PACKAGES ON PYNQ BOARD:

Example projects can be accessed from pynq website http://www.pynq.io/examples.



PYNQ community projects

PYNQ RFSoC
University Strathclyde
QPSK demo on ZCU111

PYNQ-PRIO
BYU
Partial reconfiguration Input/Output

PYNQ Hello World
Hardware accelerated image resizer example

RISC-V on PYNQ
UCSD

Extended Kalman filter
University Sydney

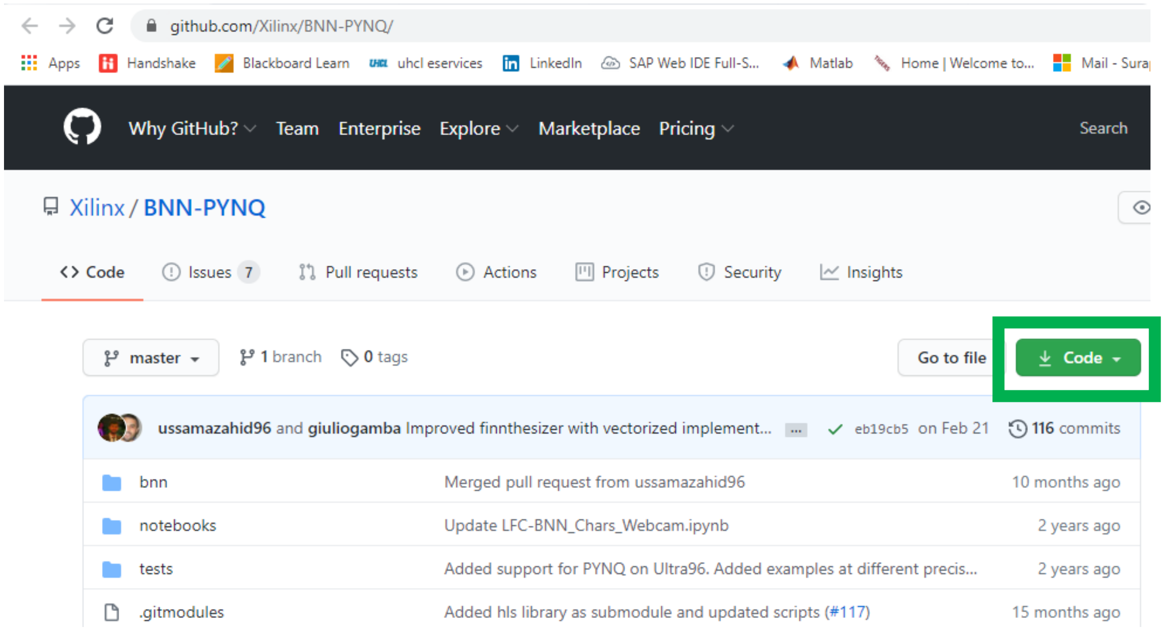
spoonNN
ETH Zurich
FPGA-based neural network inference project

iSmart DNN
FPGA-based neural network inference for DAC 2018 contest

TGIIF
FPGA-based neural network inference for DAC 2018 contest

Install the example packages on pynq board by following below steps:

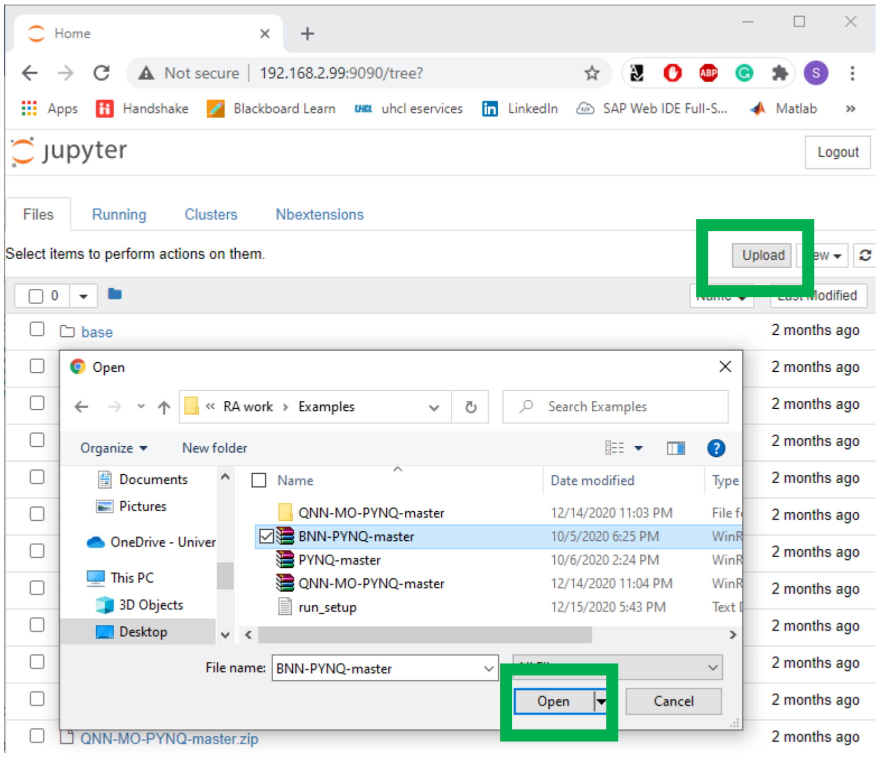
- Download the source code of the example project from github.



ussamazahid96 and giuliogamba Improved finnthesizer with vectorized implement... ✓ eb19cb5 on Feb 21 116 commits

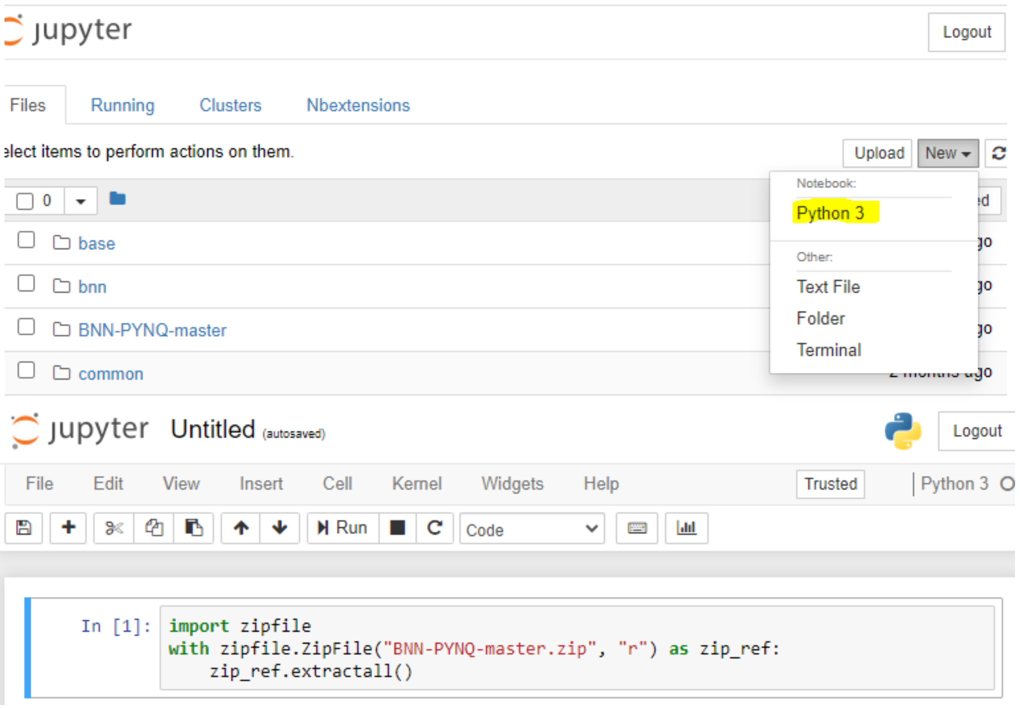
bnn	Merged pull request from ussamazahid96	10 months ago
notebooks	Update LFC-BNN_Chars_Webcam.ipynb	2 years ago
tests	Added support for PYNQ on Ultra96. Added examples at different precis...	2 years ago
.gitmodules	Added hls library as submodule and updated scripts (#117)	15 months ago

- Upload the code to jupyter notebook using UPLOAD option. Select the downloaded zip folder and click open.



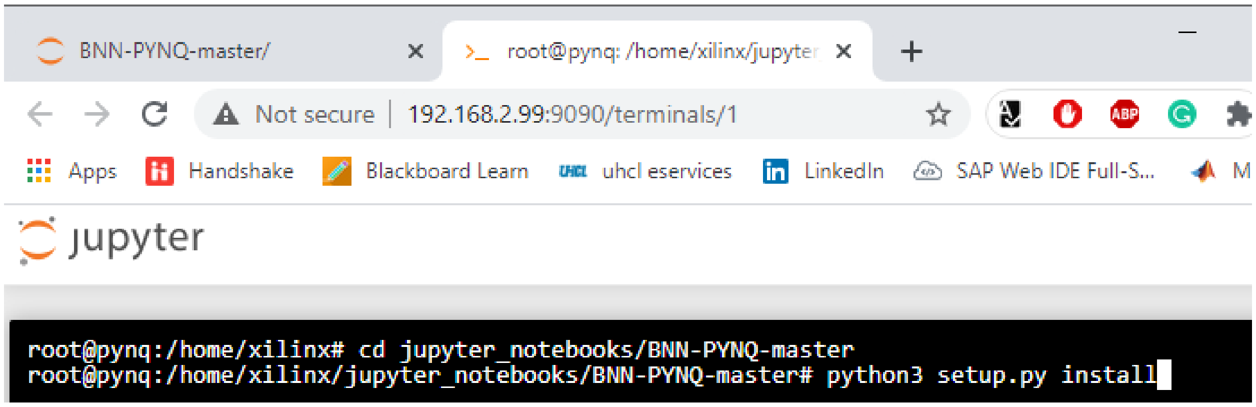
Name	Date modified	Type
QNN-MO-PYNQ-master	12/14/2020 11:03 PM	File f
BNN-PYNQ-master	10/5/2020 6:25 PM	WinR
PYNQ-master	10/6/2020 2:24 PM	WinR
QNN-MO-PYNQ-master	12/14/2020 11:04 PM	WinR
run_setup	12/15/2020 5:43 PM	Text

- Open a new notebook and unzip the file using below code:



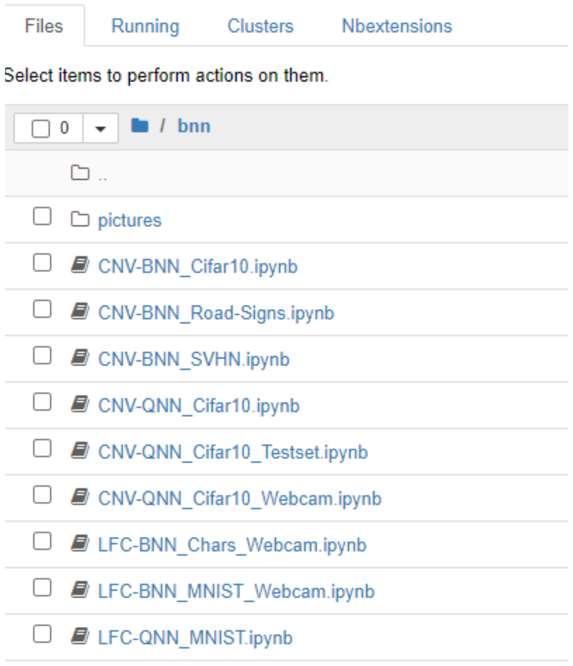
```
In [1]: import zipfile
        with zipfile.ZipFile("BNN-PYNQ-master.zip", "r") as zip_ref:
            zip_ref.extractall()
```

- Then open a new terminal in jupyter notebook and run the setup.py file using “python3 setup.py install” command.



```
root@pynq:/home/xilinx# cd jupyter_notebooks/BNN-PYNQ-master
root@pynq:/home/xilinx/jupyter_notebooks/BNN-PYNQ-master# python3 setup.py install
```

- This will install the project package on board and create a directory in the jupyter home area.



Select items to perform actions on them.

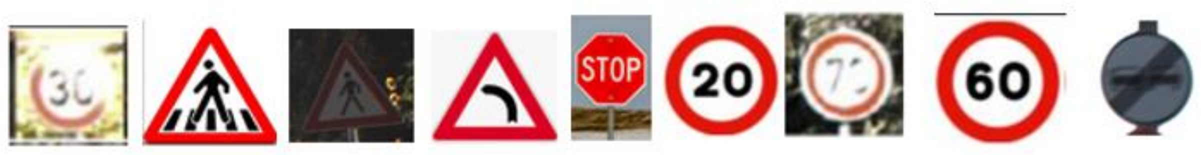
```
0  / bnn
..
pictures
CNV-BNN_Cifar10.ipynb
CNV-BNN_Road-Signs.ipynb
CNV-BNN_SVHN.ipynb
CNV-QNN_Cifar10.ipynb
CNV-QNN_Cifar10_Testset.ipynb
CNV-QNN_Cifar10_Webcam.ipynb
LFC-BNN_Chars_Webcam.ipynb
LFC-BNN_MNIST_Webcam.ipynb
LFC-QNN_MNIST.ipynb
```

EXAMPLE 1: ROAD SIGN RECOGNITION USING BNN

Source Code: https://github.com/Xilinx/BNN-PYNQ/

This example recognizes input image with a binarized neural network emphasizing 6 convolutional layers, 3 max pool layers, and 3 fully connected layers. It uses a German road-sign dataset and can classify 42 classes of road signs. The example has successfully identified the road signs from the given input images.

Input images are:



Launching BNN in hardware:

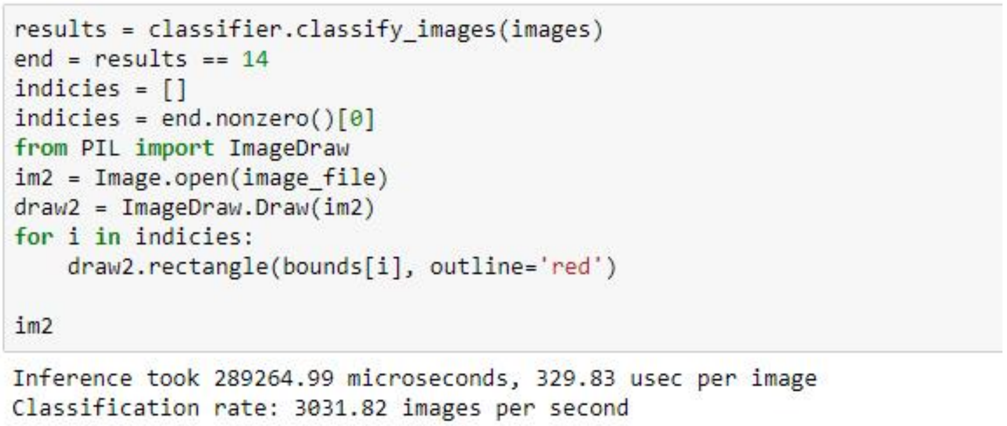
```
Inference took 4840.00 microseconds, 537.78 usec per image
Classification rate: 1859.50 images per second
Identified classes: [ 1 27 27 19 14  1  4  3 41]
Identified class name: 30 Km/h
Identified class name: Pedestrians in road ahead
Identified class name: Pedestrians in road ahead
Identified class name: Bend to left
Identified class name: Stop
Identified class name: 30 Km/h
Identified class name: 70 Km/h
Identified class name: 60 Km/h
Identified class name: End of no-overtaking zone
```

Launching BNN in Software:

```
Inference took 14312376.00 microseconds, 1590264.00 usec per image
Classification rate: 0.63 images per second
Identified classes: [ 1 27 27 19 14  1  4  3 41]
Identified class name: 30 Km/h
Identified class name: Pedestrians in road ahead
Identified class name: Pedestrians in road ahead
Identified class name: Bend to left
Identified class name: Stop
Identified class name: 30 Km/h
Identified class name: 70 Km/h
Identified class name: 60 Km/h
Identified class name: End of no-overtaking zone
```

From the results, we can clearly notice that the time taken to identify road signs in input images in hardware is less than in software.

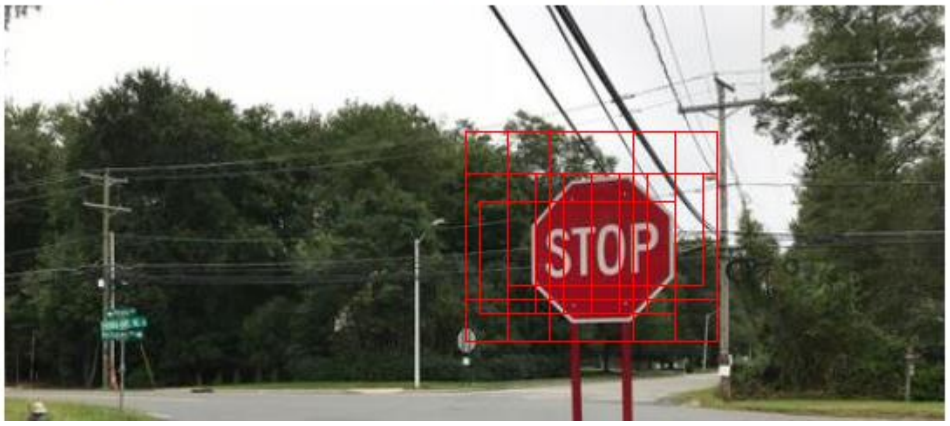
The below code snippet searches if there is any stop sign in the input image and locates if found.



```
results = classifier.classify_images(images)
end = results == 14
indicies = []
indicies = end.nonzero()[0]
from PIL import ImageDraw
im2 = Image.open(image_file)
draw2 = ImageDraw.Draw(im2)
for i in indicies:
    draw2.rectangle(bounds[i], outline='red')

im2
```

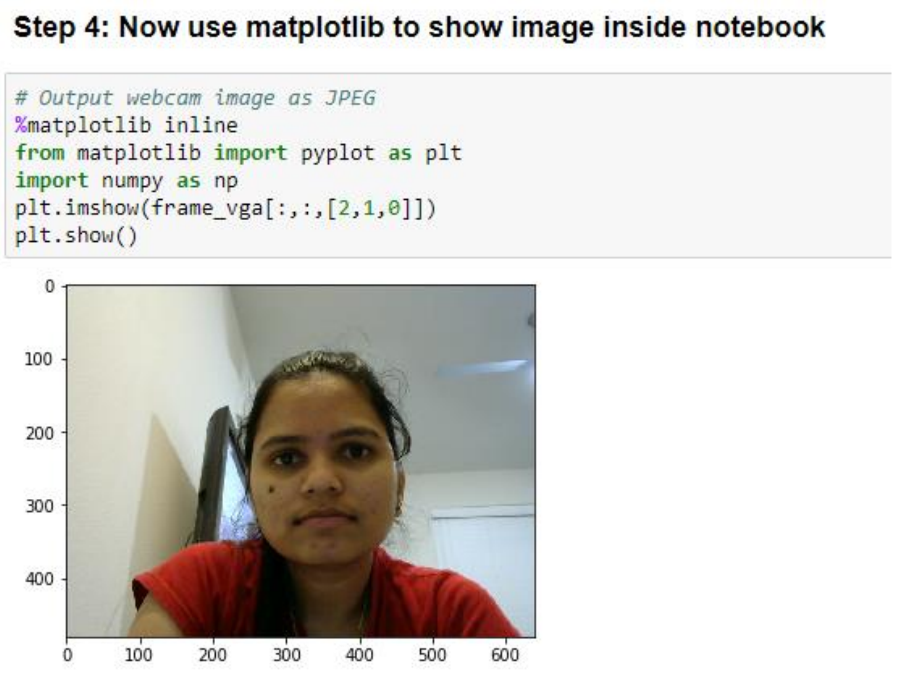
```
Inference took 289264.99 microseconds, 329.83 usec per image
Classification rate: 3031.82 images per second
```



EXAMPLE 2A: FACE DETECTION USING OPEN CV

In this example, a webcamera is connected to the USB port of the board. The webcam captures the runtime images to which open cv face detection will be applied. The example detects and locates face and eyes from the image.

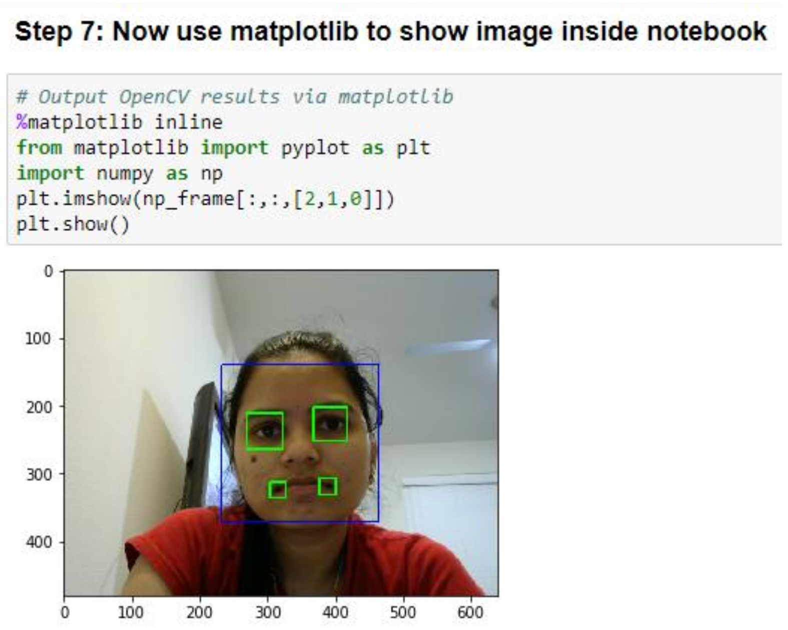
INPUT:



Step 4: Now use matplotlib to show image inside notebook

```
# Output webcam image as JPEG
%matplotlib inline
from matplotlib import pyplot as plt
import numpy as np
plt.imshow(frame_vga[:,:,[2,1,0]])
plt.show()
```

RESULT:



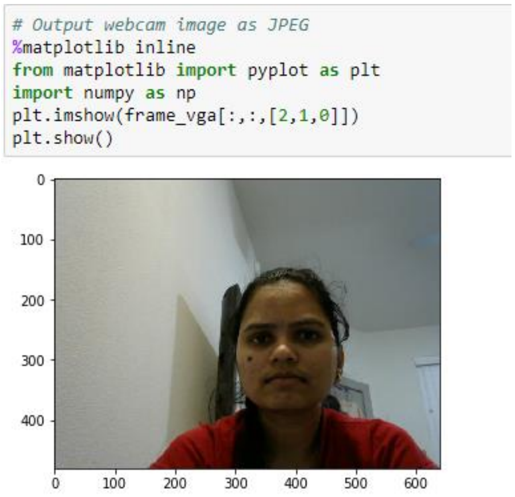
Step 7: Now use matplotlib to show image inside notebook

```
# Output OpenCV results via matplotlib
%matplotlib inline
from matplotlib import pyplot as plt
import numpy as np
plt.imshow(np_frame[:,:,[2,1,0]])
plt.show()
```

EXAMPLE 2B: EDGE DETECTION USING OPENCV

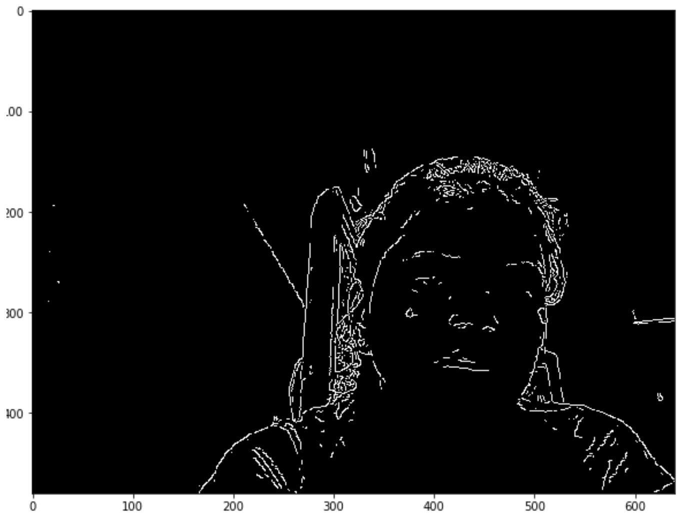
In this project, the USB camera is connected with the PYNQ USB Port. A Monitor is attached to the HDMI output port of the PYNQ board. Computer Vision-based edge detection algorithm like Canny Edge algorithm is applied to the real-time video using Python Programming. The result will be displayed on the monitor with edge detection applied.

INPUT



```
# Output webcam image as JPEG
%matplotlib inline
from matplotlib import pyplot as plt
import numpy as np
plt.imshow(frame_vga[:,:,[2,1,0]])
plt.show()
```

RESULT



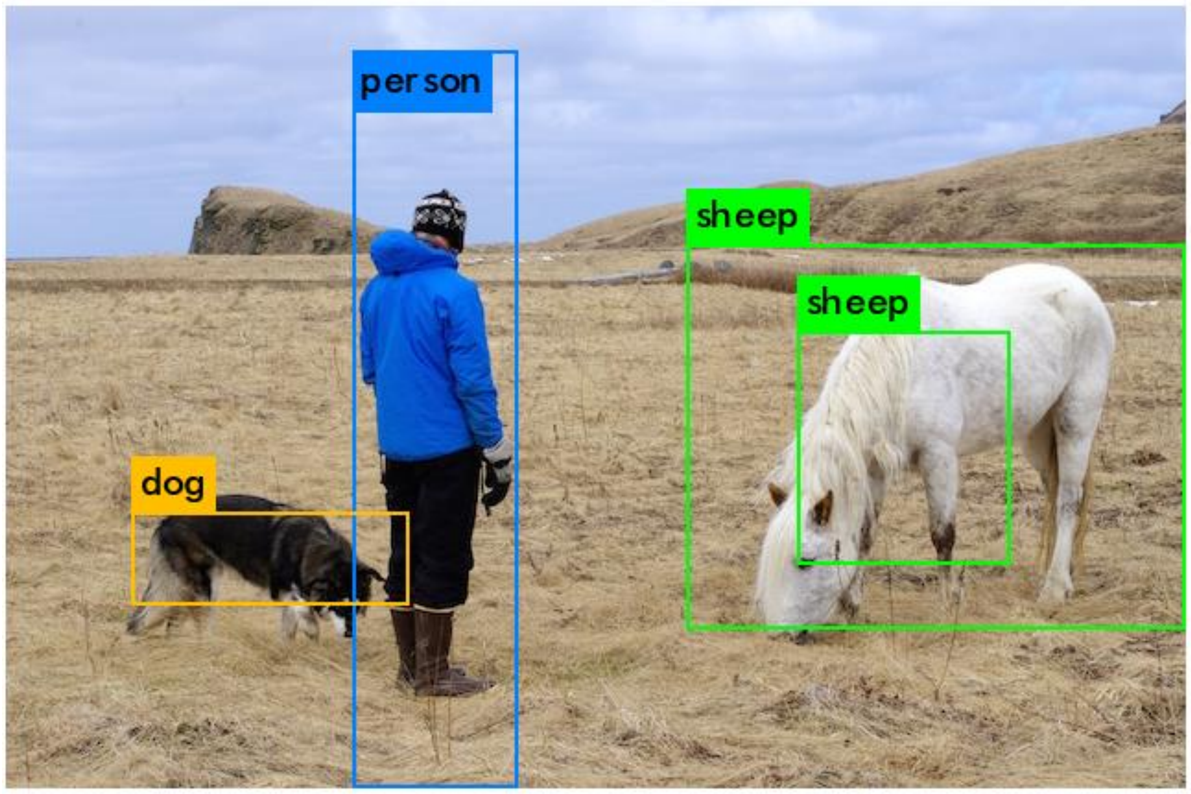
EXAMPLE 3: OBJECT DETECTION USING QNN

Source Code: https://github.com/Xilinx/QNN-MO-PYNQ

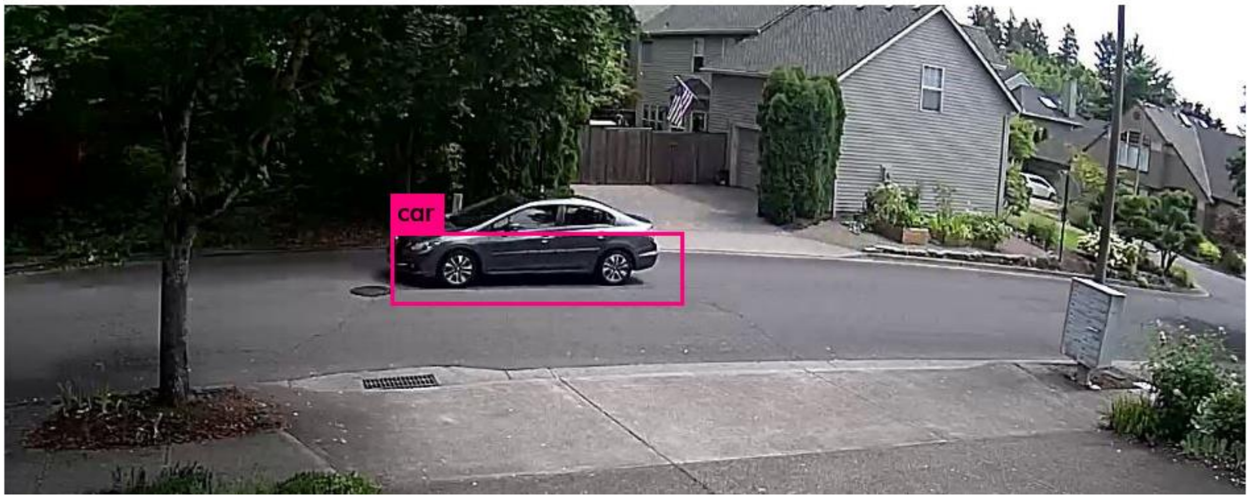
In this project, a kind of Tiny-Yolo model was developed based on the Darknet network to recognize multiple objects. Each input image is associated with a single neural network. This network works by diving the image into multiple regions and predicting bounding boxes and probabilities for each region. The object is recognized and labeled as shown below with the probability.

The project classifies multiple objects in the input image with the respective probability as shown in the below images:

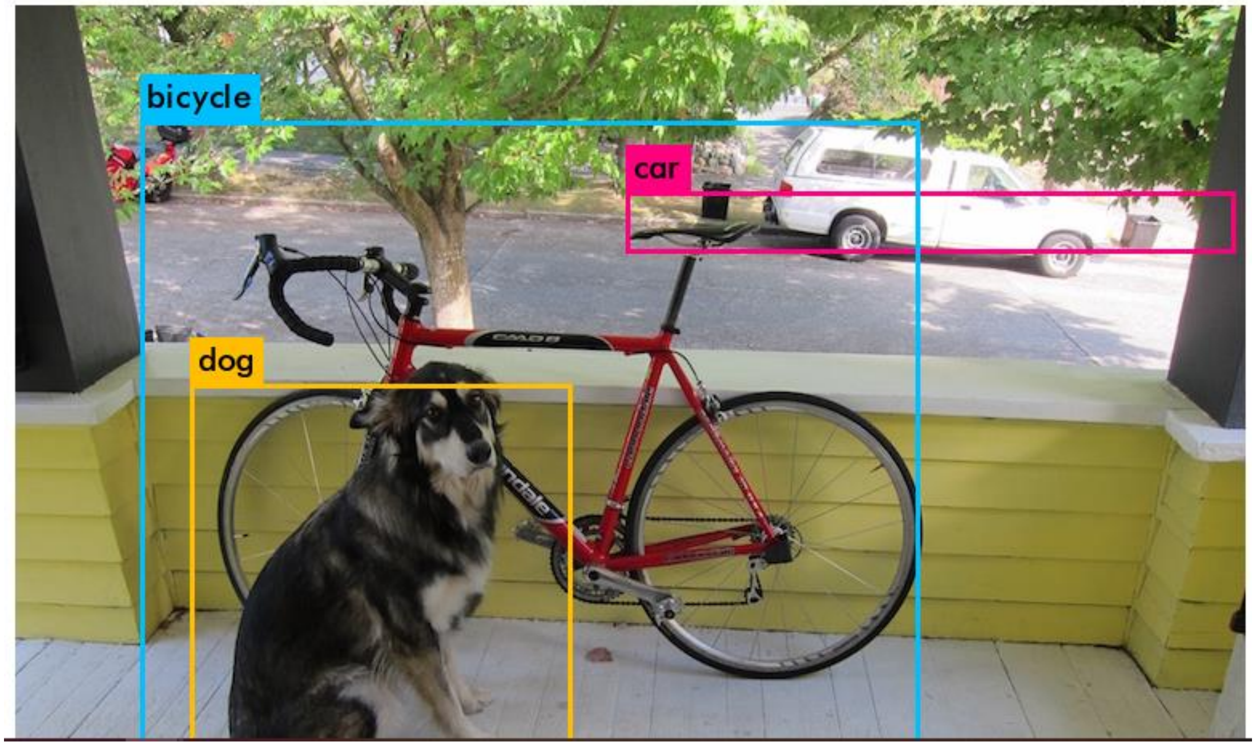
```
['person', ': 97%', 'sheep', ': 96%', 'sheep', ': 43%', 'dog', ': 85%']
```



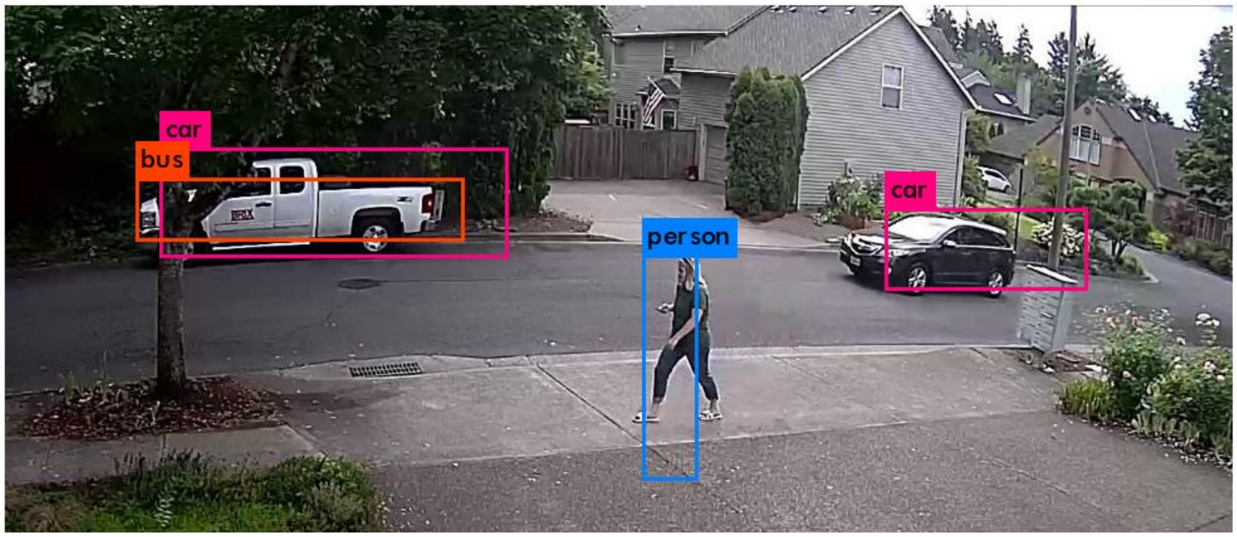
```
['car', ': 92%']
```



```
['car', ': 86%', 'dog', ': 86%', 'bicycle', ': 78%']
```



```
['car', ': 84%', 'car', ': 46%', 'person', ': 75%', 'bus', ': 30%']
```



REFERENCES:

1. Xilinx. PYNQ: Python Productivity for Zynq. url: http://www. pynq.io
2. TUL Corporation, “TUL PYNQ-Z2 Board Based on Xilinx Zynq SoC.” 2020, url: http://www.tul.com.tw/ProductsPYNQ-Z2.html.
3. BNN-PYNQ PIP INSTALL Package. url: https://github.com/Xilinx/BNN-PYNQ/
4. QNN-MO-PYNQ PIP INSTALL Package. url:https://github.com/Xilinx/QNN-MO-PYNQ
5. OpenCV Face Detection HDMI url:https://github.com/Xilinx/PYNQ/blob/v2.0/boards/Pynq-Z1/base/notebooks/video/opencv_face_detect_hdmi.ipynb
6. OpenCV Filters Webcam. Url: https://github.com/Xilinx/PYNQ/blob/v1.4/Pynq-Z1/notebooks/examples/opencv_filters_webcam.ipynb
7. OpenCV Team. About OpenCV (Open Source Computer Vision Library). url: https://opencv.org/about/
8. Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: CoRR abs/1506.02640 (2015). arXiv: 1506.02640. url: http://arxiv.org/abs/1506.02640.
9. T. Wu, Y. Wang, W. Shi and J. Lu, "HydraMini: An FPGA-based Affordable Research and Education Platform for Autonomous Driving," *2020 International Conference on Connected and Autonomous Driving (MetroCAD)*, Detroit, MI, USA, 2020, pp. 45-52, doi: 10.1109/MetroCAD48866.2020.00016.
10. Xilinx Inc., “Zynq-7000 SoC product.” 2020, url:https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html
11. V. Y. Çambay, A. Uçar and M. A. Arserim, "Object Detection on FPGAs and GPUs by Using Accelerated Deep Learning," *2019 International Artificial Intelligence and Data Processing Symposium (IDAP)*, Malatya, Turkey, 2019, pp. 1-5, doi: 10.1109/IDAP.2019.8875870.