X-LINUX-AI - object detection using Coral Edge TPU TensorFlow Lite C++ API

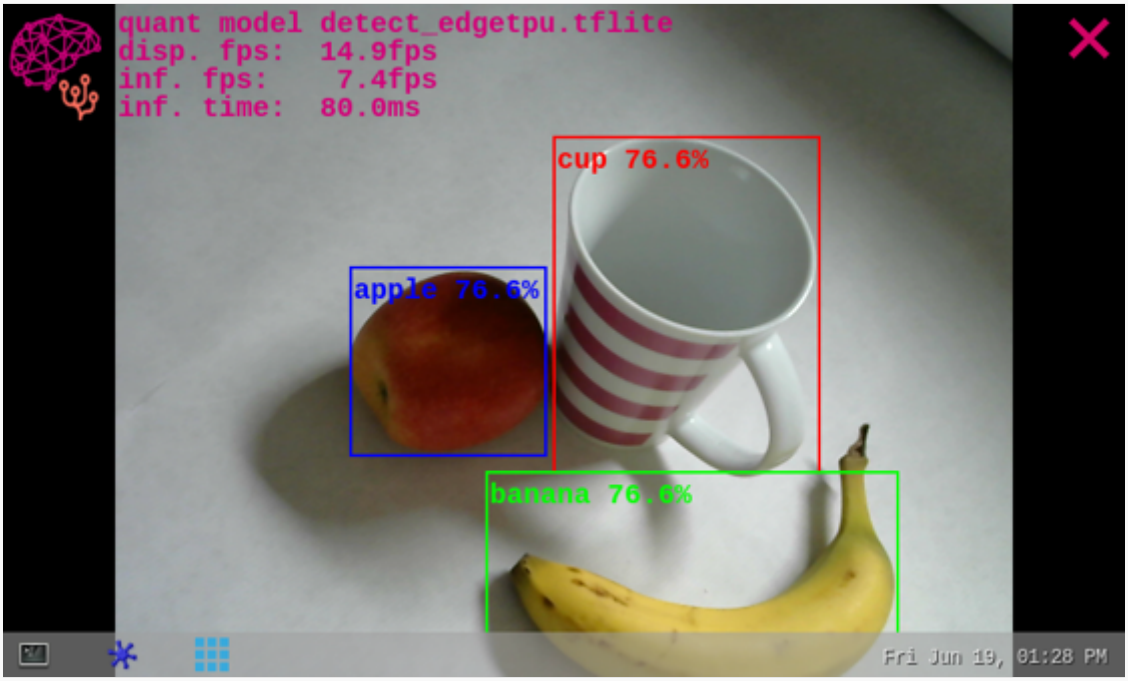
A quality version of this page, approved on *3 July 2020*, was based off this revision.

This article explains how to experiment with **Coral Edge TPU**[1] applications for object detection based on the COCO SSD MobileNet v1 model using the TensorFlow Lite C++ API.

Contents

1 Description

The **object detection**[2] neural network model allows identification and localization of a known object within an image.



C/C++ Coral Edge TPU object detection application

The application demonstrates a computer vision use case for object detection where frames are grabbed from a camera input (/dev/videox) and analyzed by a neural network model executed on the **Coral Edge TPU**[1] using the TensorFlow Lite C++ API.
A Gstreamer pipeline is used to stream camera frames (using v4l2src), to display a preview (using waylandsink) and to execute a neural network inference (using appsink).
The result of the inference is displayed on the preview. The overlay is done using GtkWidget with cairo.
This combination is quite simple and efficient in terms of CPU overhead.

The model used with this application is the **COCO SSD MobileNet v1** downloaded from the **object detection overview**[2] and converted for the Coral Edge TPU.

2 Installation

2.1 Install from the OpenSTLinux AI package repository

Warning

The software package is provided AS IS, and by downloading it, you agree to be bound to the terms of the software license agreement (SLA). The detailed content licenses can be found here.

After having configured the AI OpenSTLinux package you can install X-LINUX-AI components for this application:

```
Board $> apt-get install tflite-cv-apps-edgetpu-object-detection-c++
```

Then restart the demo launcher:

```
Board $> systemctl restart weston@root
```

2.2 Source code location

- in the Openembedded OpenSTLinux Distribution:
  <Distribution Package installation directory>/layers/meta-st/meta-st-stm32mpu-ai/recipes-samples/tflite-cv-apps-edgetpu/files/object-detection/src
- on GitHub:
  https://github.com/STMicroelectronics/meta-st-stm32mpu-ai/tree/dunfell/recipes-samples/tflite-cv-apps-edgetpu/files/object-detection/src

2.3 Re-generate the package from OpenSTLinux Distribution (optional)

Using the Openembedded OpenSTLinux distribution, you are able to rebuild the application.

Information

If not already installed, the X-LINUX-AI OpenSTLinux Distribution need to be installed by following this link

- Set up the build environment:

```
PC $> cd <Distribution Package installation directory>
PC $> source layers/meta-st/scripts/envsetup.sh
```

- Rebuild the application:

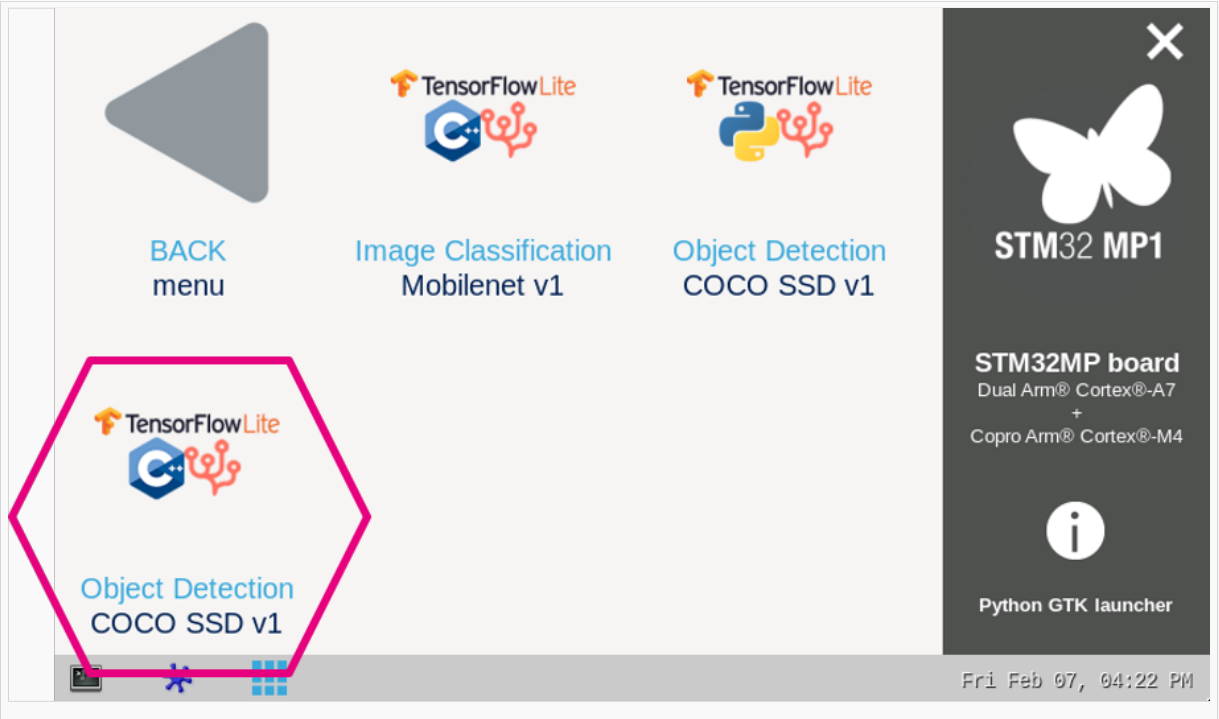
```
PC $> bitbake tflite-cv-apps-edgetpu-object-detection-c++ -c compile
```

The generated binary is available here:

```
<Distribution Package installation directory>/<build directory>/tmp-glibc/work
/cortexa7t2hf-neon-vfpv4-ostl-linux-gnueabi/tflite-cv-apps-edgetpu-object-detection-c++/1.
0-r0/tflite-cv-apps-edgetpu-object-detection-c++-1.0/object-detection/src
```

3 How to use the application

3.1 Launching via the demo launcher



3.2 Execute with the command line

The **objdetect_tfl_edgetpu_gst_gtk** C/C++ application is located in the userfs partition:

```
/usr/local/demo-ai/computer-vision/tflite-object-detection-edgetpu/bin
/objdetect_tfl_edgetpu_gst_gtk
```

It accepts the following input parameters:

```
Usage: ./objdetect_tfl_edgetpu_gst_gtk -m <model .tflite> -l <label .txt
file>


-m --model_file <.tflite file path>:  .tflite model to be
executed
-l --label_file <label file path>:    name of file containing
labels
-i --image <directory path>:          image directory with image to be
classified
-v --video_device <n>:                video device (default /dev
/video0)
--crop:                               if set, the nn input image is cropped (with the
expected nn aspect ratio) before being resized,
                                      else the nn imput image is only resized to the nn
input size (could cause picture deformation).
```

```
--frame_width  <val>:                  width of the camera frame (default is
640)
--frame_height <val>:                  height of the camera frame (default is
480)
--framerate <val>:                     framerate of the camera (default is
15fps)
--input_mean <val>:                    model input mean (default is
127.5)
--input_std  <val>:                    model input standard deviation (default is
127.5)
--help:                                show this help
```

3.3 Testing with COCO SSD MobileNet V1

The model used for test is the **detect_edgetpu.tflite** downloaded from the **object detection overview**[2] and converted for the Coral Edge TPU. If you are interested, please take a look at how this model has been converted.

Information

The different objects the neural network is able to detect are listed in the labels.txt file located in the target:

/usr/local/demo-ai/computer-vision/models/coco_ssd_mobilenet/labels.txt

To ease launching of the application, two shell scripts are available:

- launch object detection based on camera frame inputs

```
Board $>/usr/local/demo-ai/computer-vision/tflite-object-detection-edgetpu/bin
/launch_bin_objdetect_tfl_edgetpu_coco_ssd_mobilenet.sh
```

- launch object detection based on the pictures located in **/usr/local/demo-ai/computer-vision/models/mobilenet/testdata** directory

```
Board $> /usr/local/demo-ai/computer-vision/tflite-object-detection-edgetpu/bin
/launch_bin_objdetect_tfl_edgetpu_coco_ssd_mobilenet.sh
```

Information

Note that you need to populate the testdata directory with your own data sets.

The pictures are then randomly read from the testdata directory

4 References

- [1.01.1] Coral Edge TPU
- [2.02.12.2] TFLite object detection overview

Application programming interface

Central processing unit

Artificial Intelligence