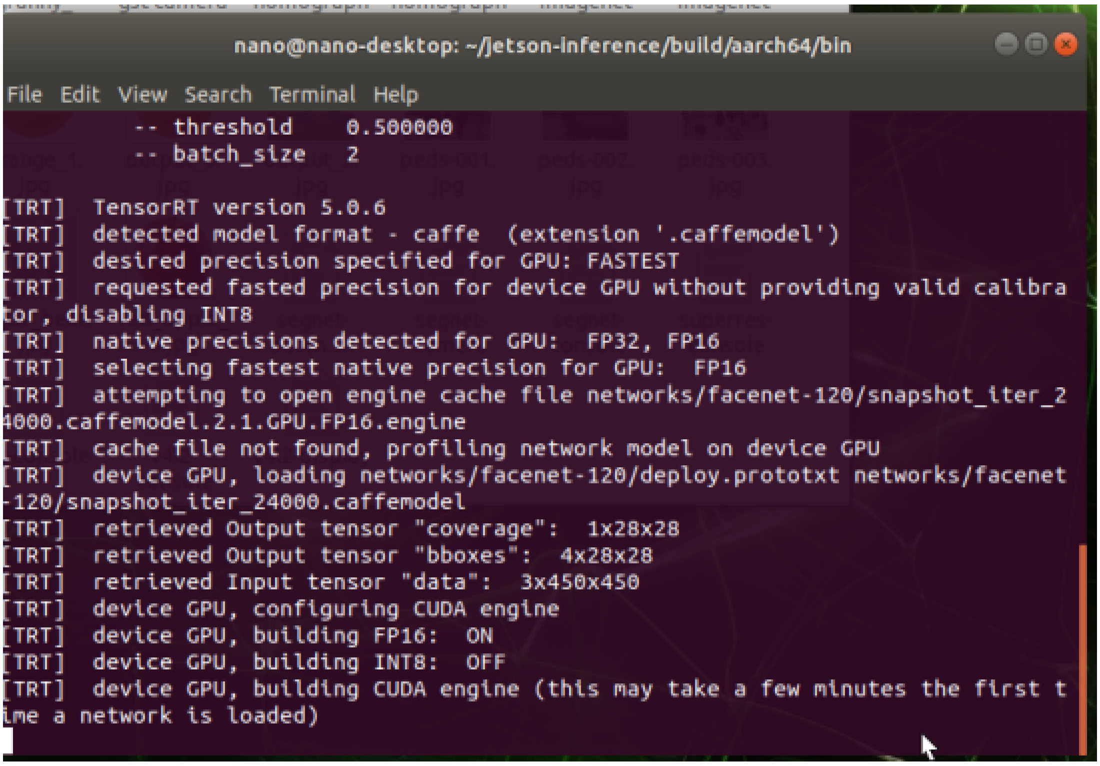
9.Using DetectNet camera Real-time detection

Run the object detection network on the real-time video source of the detectnet-camera Jetson onboard camera. Start it from the command line and the type of network you need:

```
$ ./detectnet-camera facenet          # Running using facial recognition network
$ ./detectnet-camera multiped    # Run using multi-level pedestrian/baggage
detector
$ ./detectnet-camera pednet        # Run using original single-level pedestrian
detector
$ ./detectnet-camera coco-bottle    # Detect bottle/soda can under the camera
$ ./detectnet-camera coco-dog               # Detecting dogs under the camera
$ ./detectnet-camera                     # By default, the program will run and use
```

Note: To get the best performance when running detectnet, increase the Jetson clock limit by running a script:

sudo ~/jetson_clocks.sh


nano@nano-desktop: ~/jetson-inference/build/aarch64/bin
File Edit View Search Terminal Help

```
          -- threshold     0.500000
          -- batch_size    2

[TRT]  TensorRT version 5.0.6
[TRT]  detected model format - caffe  (extension '.caffemodel')
[TRT]  desired precision specified for GPU: FASTEST
[TRT]  requested fasted precision for device GPU without providing valid calibra
tor, disabling INT8
[TRT]  native precisions detected for GPU:  FP32, FP16
[TRT]  selecting fastest native precision for GPU:  FP16
[TRT]  attempting to open engine cache file networks/facenet-120/snapshot_iter_2
4000.caffemodel.2.1.GPU.FP16.engine
[TRT]  cache file not found, profiling network model on device GPU
[TRT]  device GPU, loading networks/facenet-120/deploy.prototxt networks/facenet
-120/snapshot_iter_24000.caffemodel
[TRT]  retrieved Output tensor "coverage":  1x28x28
[TRT]  retrieved Output tensor "bboxes":  4x28x28
[TRT]  retrieved Input tensor "data":  3x450x450
[TRT]  device GPU, configuring CUDA engine
[TRT]  device GPU, building FP16:  ON
[TRT]  device GPU, building INT8:  OFF
[TRT]  device GPU, building CUDA engine (this may take a few minutes the first t
ime a network is loaded)
```

In the above execution process, each time the first execution is performed, the update model will take a long time. You need to wait patiently, when you want to use it next time, you can use it directly.

Note: By default, Jetson's on-board CSI camera will be used as the video source.

If you want to use a USB webcam,
Similar to the previous detectnet-console example, these camera applications use detection networks, except that they process live video from the camera. detectnet-camera accepts a variety of optional command line parameters, including:

- - - network flag, which changes the detection model in use (default is SSD-Mobilenet-v2).
- - - overlay flag, which can be a comma-separated combination of box, labels, conf, and none.
- The default value is --overlay = box, labels, conf display box, label and confidence values
- - - alpha sets the value of the alpha blending value to use when overriding (the default is 120).
- --threshold sets the value of the minimum detection threshold (default is 0.5).
- - - camera flag sets the camera device to be used
- Use MIPI CSI cameras by specifying the sensor index (0 or 1 etc.)
- V4L2 USB camera is used by specifying its /dev/video node (/dev/video0, , etc.).
- Default is to use MIPI CSI sensor 0 (--camera = 0)
- - - width and - - height flags set the camera resolution (default is 1280x720)
- Resolution should be set to a format supported by the camera.
- Query the available formats using:

```
sudo apt-get install v4l-utils
v4l2-ctl --list-formats-ext
```

You can combine these flags as needed, and there are other command line parameters available for loading custom models. Launch the application with the --help flag for more information, or see the Examples readme.

Here are some typical scenarios for start programs:

C ++

```
$ ./imagenet-camera       # Use GoogleNet, default MIPI CSI camera (1280 × 720)
$ ./imagenet-camera - - network = facenet    # Use RESNET-18, default MIPI CSI camera (1280×720)
$ ./imagenet-camera - - camera = /dev /video1    # Use GoogleNet, V4L2 camera / dev/video1 (1280x720)
$ ./imagenet-camera - - width = 640 - - height = 480    # Use GoogleNet, default is MIPI CSI camera (640x480)
```

Python

```
$ ./imagenet-camera.py    # Using GoogleNet, the default MIPI CSI camera (1280x720)
$ ./imagenet-camera.py - - network = facenet    # Use RESNET-18, the default MIPI CSI camera (1280x720)
$ ./ imagenet-camera.py - - camera = /dev/video1     # Use GoogleNet, V4L2 camera /dev/video0 (1280x720)
$ ./imagenet-camera.py - - width = 640 - - height = 480    # Use GoogleNet, default is MIPI CSI camera (640x480)
```

Visualization

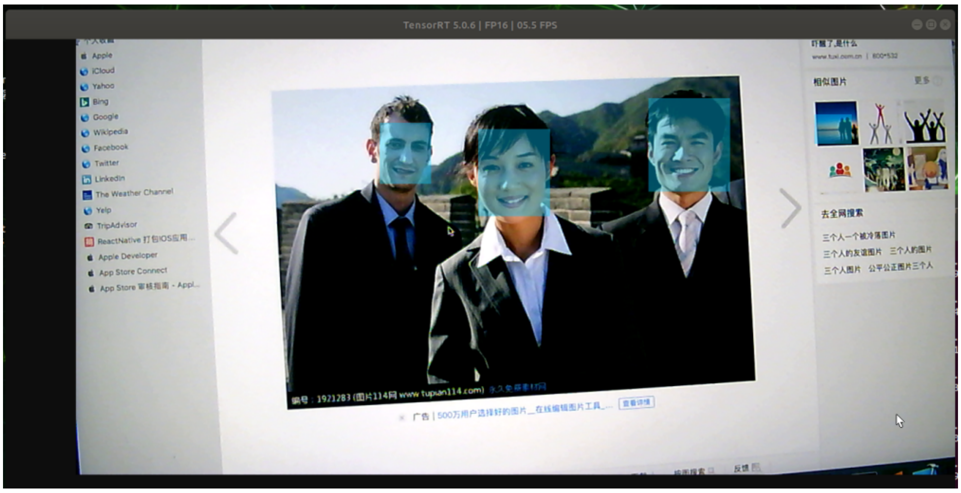
The OpenGL window displays a real-time camera video stream, which covers the bounding box of the detected object. Please note that the current SSD-based models have the highest performance.

This is the one using this coco-dog model:

```
# C ++
$ ./Detectnet-camera - - network = coco-dog
#Python
$ ./Detectnet-camera.py - - network = coco-dog
```

If the desired object is not detected in the video feed, or if you get false detection, try using the - - threshold parameter to lower or increase the detection threshold (the default is 0.5).

After executing the first command,we can detect multiple faces. As shown below.


TensorRT 5.0.6 | FP16 | 05.5 FPS